RPC-2350 USER'S MANUAL

TRADEMARKS

CAMBASIC™and PC SmartLINK™are trademarks of Octagon Systems Corporation.

Microsoft® BASIC is a trademark of Microsoft Corporation.

Microsoft® Windows®, Windows 95®, and Windows 98® are trademarks of Microsoft Corporation.

Windows Terminal is registered with Microsoft Corporation.

HyperTerm is copyright by Hilgraeve Inc. and is developed for Microsoft Corporation.

Procomm is copyright by Datastorm Technologies, part of Symantec Corporation

Remote Processing Corporation
7975 E. Harvard Ave.
Denver, Co 80231
Ph. : (303) 690 1588
Fax: (303) 690 1875
www.rp3.com
info@rp3.com

NOTICE TO USER

The information contained in this manual is believed to be correct. However, Remote Processing assumes no responsibility for any of the circuits described herein, conveys no license under any patent or other right, and make no representations that the circuits are free from patent infringement. Remote Processing makes no representation or warranty that such applications will be suitable for the use specified without further testing or modification. The user must decide fitness for a particular use.

Remote Processing Corporation's general policy does not recommend the use of its products in life support or applications where the failure or malfunction of a board may threaten life or injury. Install redundant or backup safety systems as appropriate to the application.

FCC AND EMI NOTICE

The RPC-2350 and RPC-2350G is intended as an OEM product in an industrial environment. It was not tested for EMI radiation. When operated outside a suitable enclosure, the board and any cables coming from the board will radiate harmful signals that interfere with consumer and industrial radio frequencies. It is your responsibility properly to shield the RPC-2350/2350G and cables coming from it to prevent such interference.

P/N 1761
Revision: 1.4

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER 1 OVERVIEW

CHAPTER SYNOPSIS

● Brief description of the RPC-2350 and RPC-2350G
● How this manual is organized
● How to get technical support
● Application disk

DESCRIPTION

The RPC-2350 is an embedded controller with a built in Basic language. The RPC-2350G also includes a graphics display interface. Other features include:

♦ Built in CAMBASIC programming language autoruns at power up. On card Flash EPROM programmer saves programs.

♦ Eight single ended or 4 differential analog inputs convert voltage inputs to a digital value using a 12 -bit (4096 count) A/D converter. Two 12-bit analog or 4-20 mA. outputs also available.

♦ Keypad port for an operator interface. The 16 position keypad is automatically scanned and is read using the KEYPAD command.

♦ Two RS-232 serial ports are programmable for baud rate, parity, length, and stop bits. Both inputs and outputs have a 256-byte buffer.

♦ A watchdog timer resets the card if the program "crashes." The timer is enabled and disabled by software.

♦ 48 general purpose digital I/O lines, 8 of which are high current outputs. These lines can connect to another opto rack.

♦ 128K of battery backed RAM is standard. A 512K RAM optionally available.

♦ Built in EPROM programmer save programs for autorun on power up or reset.

The RPC-2350 uses a Z8S180 CPU operating double clocked at 18 MHZ. Double clocked operation makes it twice as fast as its equivalent 64180 CPU. The board operates stand alone or on an RS-485 network. Its 5.0" x 8.0" size makes it easy to mount in a NEMA box.

CAMBASIC programming language is standard. This language is similar to Microsoft GW Basic. All hardware is supported by Basic commands. A complete description of CAMBASIC commands is in the *CAMBASIC Programming Guide.*

Program development can take place on your PC, using your word processor, or on the RPC-2350. Programs from your PC can be downloaded using PC SmartLink or other serial communication program.

Customization
We can customize the RPC-2350 for your production needs. Some examples include latching connectors, installing a specific combination of memory, soldering IC's directly to the board (where practical), and removing components for cost reductions. You must purchase at least 25 boards and allow for extra lead time. We can provide you with a firm quote ahead of time.

MANUAL ORGANIZATION

This manual provides all the information required to install, configure, and use the features on the RPC-2350. The manual is organized by function. The first section deals with getting the board operational. Other sections address on board I/O and capabilities.

This manual makes references to the RPC-2350. The RPC-2350 and RPC-2350G are very similar. Unless otherwise noted, everything that applies to the RPC-2350 also applies to the RPC-2350G.

This manual assumes you are familiar with some type of BASIC programming software. The syntax used by CAMBASIC is similar to Microsoft's GW or QuickBASIC. If you are not experienced with BASIC software, you may want to refer to books and training programs available through your local software store. The *CAMBASIC Programming Manual* has information and examples for all commands.

NOTE: The RPC-2350 uses a Zilog Z8S180 processor. Additional information can be obtained from Zilog at www.zilog.com or your local representative.

MANUAL CONVENTIONS

The RPC-2350 and RPC-2350G are very similar. Unless otherwise noted, references to the RPC-2350 also apply to the RPC-2350G. The primary differences between the two boards are power input and graphics display capability.

Information appearing on your screen is shown in a different type.

Example:

```
CAMBASIC (tm) V1.4
© 1985-94 Octagon Systems Corporation
© 1999 Remote Processing Corporation
All rights reserved - free 32434
```

Symbols and Terminology

NOTE: Text under this heading is helpful information. It is intended to act as a reminder of some interaction with another part of the manual or device that may not be obvious.

WARNING:
Information under this heading warns you of situations that might cause catastrophic or irreversible damage.

Wn[-]	Denotes jumper block pins.
< xxx>	Paired angle brackets are used to indicate a specific key on your keyboard. For example < esc> means the escape key.

BASIC uses decimal convention for designating addresses and data. There are times, however, when hexadecimal notation is more convenient to use.

The hexadecimal notation used in this manual and by CAMBASIC is the ampersand character (&) before the number. A &8C stands for 8C hexadecimal

DEMONSTRATION DISK

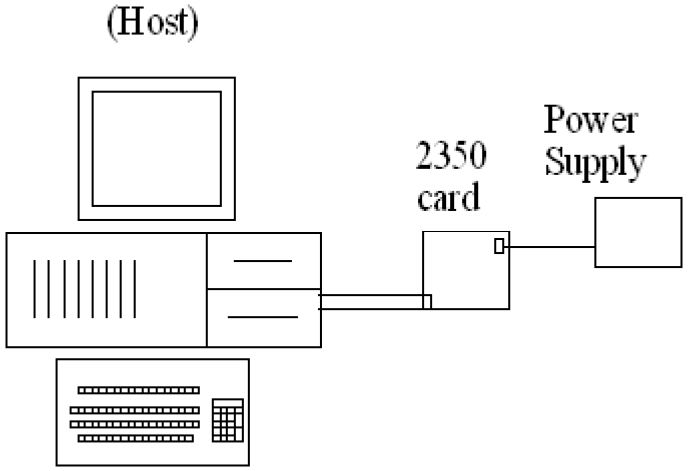
A disk with many sample programs is included with this manual. This manual references programs on the disk. You may use the code on these disks in your own CAMBASIC program. These programs are for demonstration purposes only and there is a good probability you will modify them to suit your needs (i.e., safety checks).

TECHNICAL SUPPORT

If you have a question about the RPC-2350 or CAMBASIC used on it and cannot find it in this manual, call us and ask for technical support.

When you call, please have your RPC-2350 and CAMBASIC manuals ready. Sometimes knowing what the RPC-2350 is used for is helpful, so please be ready to describe its application and the problem.

Phone: 303-690-1588
FAX: 303-690-1875
E-mail: info@rp3.com



CHAPTER 2 SETUP AND OPERATION

CHAPTER SYNOPSIS

- Running the RPC-2350 for the first time
- How to run under DOS and Windows environments
- Uploading and downloading programs
- Handy programming techniques
- Developing in Windows
- Troubleshooting if it does not work right away

INTRODUCTION

The RPC-2350 and RPC-2350G are very similar. The major difference is the RPC-2350G has graphics control software and hardware. The memory maps between the two boards are different.

This manual will refer to the RPC-2350. Unless otherwise expressed, the same information applies to the RPC-2350G.

The RPC-2350 is ready to program when you connect it to a terminal or PC and apply power. This chapter describes what is needed to get a sign on message and begin programming.

Requirements for uploading and downloading programs are discussed. A "Where to go from here" section directs you to the chapters to read to use the various capabilities of the RPC-2350 Finally, a troubleshooting section helps on the most common problems.

OPERATING PRECAUTIONS

The RPC-2350 is designed to handle a wide temperature range and operating conditions. These characteristics require using CMOS components. CMOS are static sensitive. To avoid damaging these components, observe the following precautions before handling the RPC-2350.

1. Ground yourself before handling the RPC-2350 or plugging in cables. Static electricity can easily arc through cables and to the card. Simply touching a metal part on your PC can greatly reduce static.

2. Do not insert or remove components when power is applied. While the card is a + 5 volt only system, other voltages are generated on the card.

EQUIPMENT

You will need the following equipment to begin using the RPC-2350:

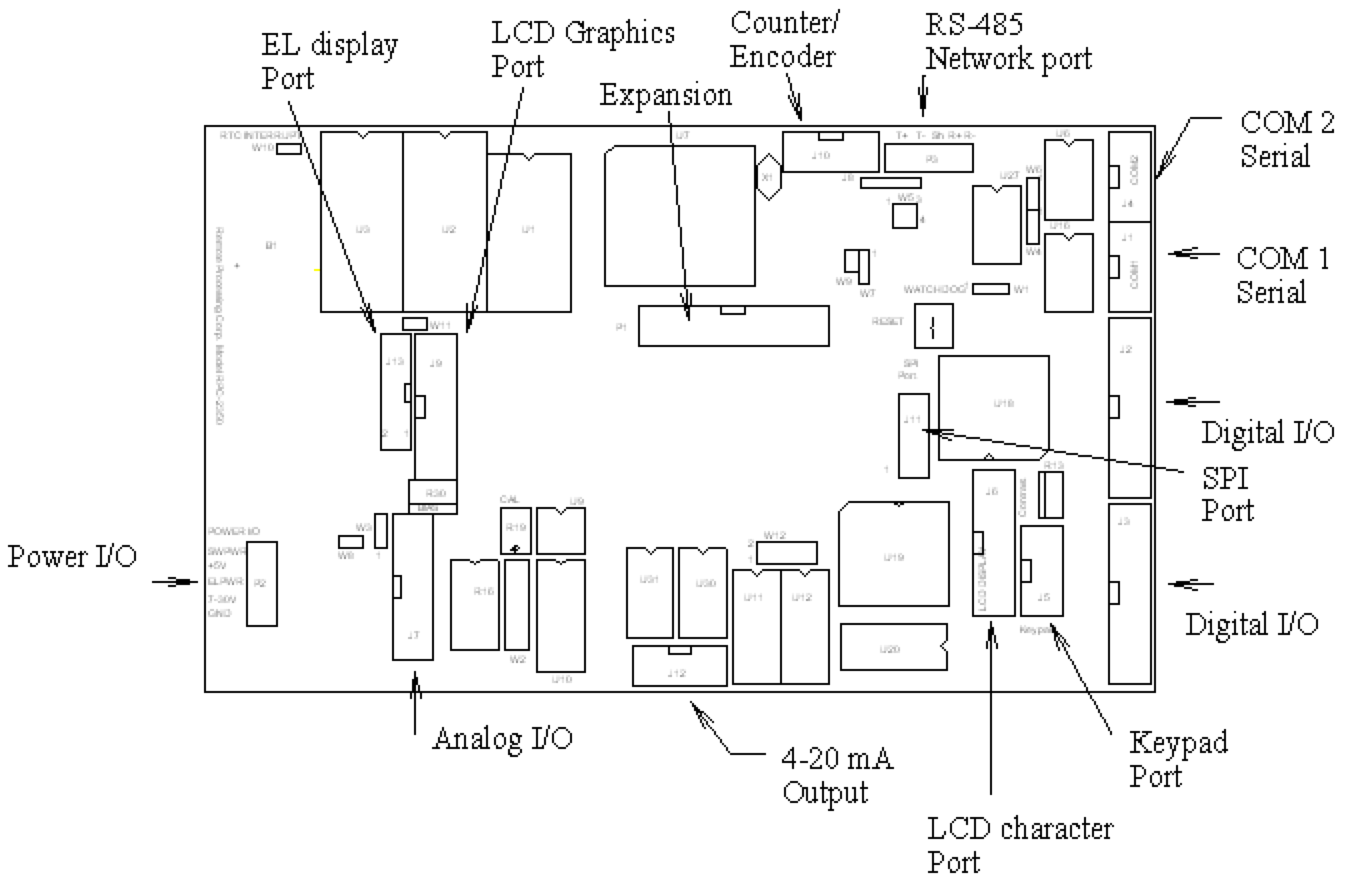
RPC-2350 embedded controller
A PC with a serial port and communications program
VTC-9 serial cable
+ 5, 300 mA. power supply or
7 to 30V applied to "7-30V" terminal on P2.

Do not connect any accessory components, such as a display or keypad, to the board until you are sure the board works in your environment.

The *CAMBASIC Programming Manual* is strongly recommended. It can be downloaded free at our web site (www.rp3.com). Refer to *Chapter 4 Serial Ports* for wiring information to make your own cable.

COMMUNICATION PROGRAM

A terminal program is used to talk with the RPC-2350



board. All programming and initial communication between the RPC-2350 and outside world is done through RS-232.

The terminal program you use depends upon the operating environment. The vast majority of environments are PC's operating either DOS or Microsoft Windows. If you are using another operating environment, then just read "All terminal programs" below and set up your terminal program accordingly.

All terminal programs
Set communication parameters to

Baud rate:	19200
Data length:	8
Parity:	none
Stop bits	1

The RPC-2350 does not send a CTS signal on its communication port. If your terminal or communications software requires this or other signals (DCD, DSR), you may have to tie them to the appropriate levels. You can probably ignore these lines in software. Windows Terminal and Hyper Terminal, PC SmartLink, or Procomm does not need them.

Set "handshake" control to "none" or OFF as appropriate.

Default communication method between the RPC-2350 and PC is RS-232 ASCII. This includes file upload and download. Other protocols such as XMODEM are not used.

A terminal program should be able to recognize a 'prompt' character. A prompt character is used to control a transfer when sending a file (as is done during a program download). When the RPC-2350 sends a '> ' prompt character, it is saying "OK to send the next line." Standard Windows HyperTerm does not allow this. Windows Terminal, Procomm and PC SmartLink do.

If your terminal program does not allow for prompt characters during text transfers, next best is to put a delay between sending lines (as is done in HyperTerm and Terminal). How much depends upon what you are sending. If you are downloading a small program (< 1000 lines), then 1/10 second is plenty of time. If you are downloading data files, then you may need more time, depending upon the complexity. The major penalty when downloading with delays is increased download time.

DOS mode
Start your serial communication program (PC SmartLink, Procomm, or other). Set the COM port to the one you will be using.

Windows mode
You may use either Windows 3.1 Terminal or Windows 95 HyperTerm communication programs. For a fast setup, use one of the following files:

CAMBASIC.TRM	Windows terminal
CAMBASIC.HT	Hyper terminal

Make sure you set the COM port number under "Properties."

See "Developing programs in Windows" later in this section for more information on writing and editing programs.

FIRST TIME OPERATION

Make sure your terminal is set up as described above.

Become familiar with the locations of the connectors before getting started. See Figure 2-1.

RPC-2350 jumpers have been set at the factory to operate the system immediately. For first time operation, do not install any connectors or parts unless specified below. Jumpers should be kept in default positions.

1. The **RPC-2350** requires + 5 ±0.25 VDC at 300 mA.

   The **RPC-2350G** can use + 5V or 6.5 to 30 VDC at 150 to 320 mA.(Current draw depends upon supply voltage. More voltage = less current).

   Make sure power is off. Connect the power supply to the appropriately marked terminals on the RPC-2350 or RPC-2350G.

   + 5V is connected to P2, "+ 5V". Make sure jumper W8 is not installed.

   7 to 30 VDC is connected to P2, "7-30V" (RPC-2350G only). Make sure jumper W8 is installed.

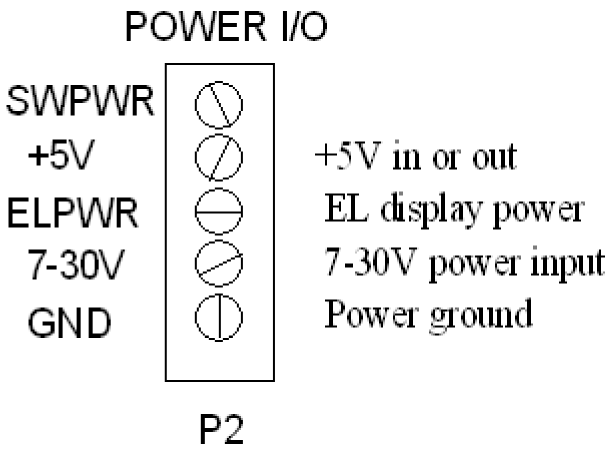


Figure 2-2 Power connector detail

Ground is connected to "GND"

"+ 5V" can be power input or output. When W8 is installed on the RPC-2350G, it is power output. When W8 is not installed, it is power input. See Chapter 16 for more information.

"SWPWR" is a high current switch to ground. See Chapter 6, *High current output at P2* for more information.

2. Connect one end of the VTC-9 connector to the 10 pin COM1 (programming) port on the RPC-2350. Refer to Figure 2-1 for connector location.

   Connect the VTC-9 serial cable to the PC's COM1 or COM2 port .

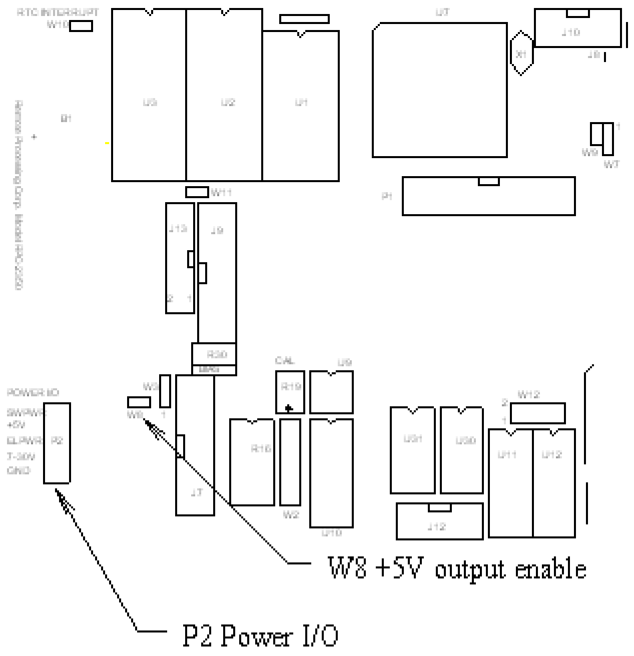


Figure 2-3 Power and jumper location detail

3. Start your terminal program (if not running already).

4. Turn on your power supply. On power up a copyright message is printed.

```
CAMBASIC (tm) V1.4
(c)1985-94 Octagon Systems Corporation
(c)1999  Remote Processing Corporation
All rights reserved - free 32434
```

   If a nonsense message appears, your terminal or PC may not be set to the appropriate communication parameters. If the system still does not respond, refer to "TROUBLESHOOTING" later in this chapter.

4. The system is now in the "immediate mode" and is ready for you to start programming. Type the following program (in upper or lower case:

```
10 FOR X = 0 TO 2
20 PRINT "  Hello ";
30 NEXT
40 PRINT
```

   Now type **RUN**

   The system will display:

```
Hello Hello Hello
```

UPLOADING AND DOWNLOADING PROGRAMS

Downloading programs means transferring them from your computer to RAM on the RPC-2350. Uploading means transferring programs from RAM back to your computer. This section explains how to do both of these procedures using PC SmartLink and both Windows terminal programs. Generalized instructions for other terminal programs are given at the end of this chapter.

Uploading - Windows
In the previous section, you wrote a test program. To upload that program to a PC and save it to disk, first type "list" but do not hit the < Enter> key.

Select "Transfer" then "Capture to disk". Enter the name you want to save it as.

After that window closes, hit the enter key. The program will be listed and captured by the terminal program. To stop capture, select "Transfer" then "Capture to disk" again. Then select stop transfer. Your program is now saved.

Uploading - SmartLink
In the previous section, you wrote a test program. To upload that program to a PC and save it to disk:

1. Press the < F1> key. A window with the main menu will appear.

2. Press the letter U (upper or lower case). Your program will begin to transfer from RAM to the PC. When menu appears.

3. To save a program to disk, type the letter S. You are prompted for a file name. Enter the file name you want the program saved under.

4. Press < F2> to return to the immediate mode.

NOTE: Some versions of PC SmartLink have pull down menus or will operate differently. Refer to the SmartLink manual for the version you are using.

Downloading - Windows
Select "Transfer" then "Send Text file". Select the program. Downloading begins.

For test purposes, select one of the programs on the RPC-2350 demo disk.

Depending upon the program (Terminal or HyperTerm) you will see progress on your screen in different ways.

You may need to change the amount of delay between lines. This is set under "Properties", "ASCII Setup". 1/10 second is usually adequate for all program sizes. You can set it to 10 milli-seconds if your program is small (under 500 lines). As the program gets larger, this time should increase. A result of a short delay time is missing or garbled program lines.

Downloading programs - SmartLink
To practice downloading a program, type

```
NEW<return>
```

Perform the following when using PC SmartLink:

1. Press the < F1> key to view the main menu.

2. SmartLink has a buffer which is used to temporarily store the program. If you followed these instructions without exiting SmartLink, the previously uploaded program is in the buffer and may be downloaded. However, lets assume you just started SmartLink. Press the L key to get the program from the disk.

3. Enter the filename to get the file.

4. Press D to download the program.

5. Press the < F2> key to return to the program. You can list the program by typing:

   ```
   list
   ```

   or

   ```
   /
   ```

Other communications software
The following is general information when using another terminal emulation program (Procomm, etc.).

When uploading or downloading files, select ASCII text format. Other formats are not used.

CAMBASIC does not know when you are typing in a program or if something else (laptop or mainframe) is sending it characters. The upload and download file does not contain any special control codes, it is simply ASCII characters.

Uploading programs is simply a process of receiving an ASCII file. You or your program simply needs to send "LIST" to receive the entire program.

Downloading a program requires transmitting an ASCII file. CAMBASIC is an incremental line compiler. As you type in (or download) a line, CAMBASIC compiles that line. The time to compile a line depends upon its complexity and how many line of code have been entered.

CAMBASIC must finish compiling a line before starting the next one. When a line is compiled, a "> " character is sent by the card. This should be your terminal programs pacing character when downloading a program.

If your communications program cannot look for a pacing prompt, set it to delay transmission after each line is sent. A 100 ms delay is usually adequate, but your CAMBASIC program may be long and complex and require more time. A result of a short delay time is missing or garbled program lines.

COM1 on the RPC-2350 does not recognize the CTS or RTS lines. The CTS line is pulled high on the RPC-2350. The effect of not recognizing these lines is your PC or terminal cannot hold off the RPC-2350's transmission. Converse, the RPC-2350 cannot hold off the host from sending it data.

DEVELOPING PROGRAMS IN WINDOWS

Programs can be completely written and downloaded using programs normally available in Microsoft Windows. The two programs you need are a terminal program (HyperTerm or Terminal) and an editing program (Notepad, Wordpad, Word, etc.). Both programs can be open at the same time, making editing and changing just that much easier. Just switch between the two programs (using < Alt> -< Tab> or clicking the task bar).

Versions of Word will not let the terminal program open the file you are working on. You actually have to close the file in Word before you can download it.

The following setup files are on the 2350 applications disk. Use these to quickly set up your terminal.

CAMBASIC.TRM	Windows terminal
CAMBASIC.HT	Hyper terminal

You may have to change the COM port to match your system. These set up files put the terminal in the following configuration:

Baud:	19200
Parity:	none
Data:	8 bits
Stop:	1 bit
Flow:	None
Line delay:	0.01 sec

Do NOT use FIFO buffers in HyperTerm.

Editing, downloading, and pasting are easy using Notepad or Wordpad. These programs allow you to edit a program, save it, then download it using HyperTerm. Download code using text transfers.

To change a portion of code, you can make the change then copy the new line. In HyperTerm, select Edit, then Paste to Host and the new line will be sent. Terminal does not have this feature. If you are writing code without line numbers, be sure to put the line number in first. In all cases, you will have to hit < Enter> to finish the line.

When you save a program, be sure to save it as a text file. Wordpad tends to be a little more annoying than Notepad because it always asks you what kind of file to save it as. Notepad does not. These programs may append a .txt extension to the file name. This is OK.

When using the line edit feature in CAMBASIC you may see some extra characters and numbers on the display. Try to ignore them as best you can.

EDITING PROGRAMS AND HINTS

Files uploaded or downloaded are simply ASCII DOS text files. No special characters or control codes are used. You may create and edit programs using your favorite word processor or editor. Just be sure to save files in DOS text format.

There are two ways you can write CAMBASIC programs: With or without line numbers. Even this is not hard and fast as you can write using a mixture.

You can write a program in lower case characters. CAMBASIC translates them to upper case.

Some programmers put "NEW" as the first line in the file. During debugging, it is common to insert "temporary" lines. Adding NEW ensures that these lines are gone. Downloading time is increased when the old program is still present.

Instead of uploading and downloading programs, you can save them to the on card Flash EPROM. This is useful if you are using a terminal to write programs. Make sure the 'Autorun' jumper is installed (See *Chapter 3 SAVING PROGRAMS*). To prevent automatic program execution on power up, insert the STOP statement at the beginning of the program (such as line 1). When you power up the RPC-2350, the program is transferred into RAM and executed. Delete the program line with the STOP statement to normally start programs.

Writing with line numbers
You can manually enter line numbers. The problem is when you have to add another line. It is quite common for programs to grow and run out of line numbers. You can execute the RERUM command to renumber lines.

A technique used to further program documentation and reduce code space is the use of comments in a downloaded file. For example, you could have the following in a file written on your editor:

```
'Check VAT temperature
'Read the output from the RTD and
' calculate the temperature

2200 a = ain(0) :'Get temp
```

The first 3 comments downloaded to the RPC-2350 are ignored. Similarly, the empty lines between comments are also ignored. Line 2200, with its comment, is a part of the program and could be listed. The major penalty by writing a program this way is increased download time.

NOTE: Some versions of PC SmartLink may optionally strip comments before downloading. Check your manual to see if this option is available.

If you like to write programs in separate modules, you can download them separately. Modules are assigned blocks of line numbers. Start up code might be from 1 to 999. Interrupt handling (keypad, serial ports) might be from lines 1000 to 1499. Display output might be from 1500 to 2500. The programmer must determine the number of lines required for each section.

When replacing a program or section, downloading time is increased. Blocks of line numbers cannot be renumbered by CAMBASIC when other parts of the program are installed. However, if a particular section is the only program downloaded, then line renumbering in that range is possible. Refer to the CAMBASIC RENUM command.

CAMBASIC automatically formats a line for minimum code space and increased readability. For example, you could download the following line of code:

```
10 fora=0to5
```

When you listed this line, it would appear as:

```
10 FOR A = 0 TO 5
```

Spaces are initially displayed but not stored. The following line:

```
10 for a    =    0    to    5
```

would be compressed and displayed as in the second example above. Spaces are removed.

Writing without line numbers
Many programming languages such as C and versions of BASIC and do not use line numbers. CAMBASIC uses line numbers simply because you can edit them through a serial terminal.

You can write and edit CAMBASIC programs without using line numbers. This makes for much more readable code.

The demo program NOLINES.BAS is such a program. (Other program demos may also not use line numbers.) The key is to use the AUTO command. This command automatically assigns a number to each program line as it is entered.

There are a few things you have to keep in mind.

1. Put the NEW statement at the start of the program file. This removes old code.

2. You must use the remark statement (') in lines that do not have any code. This is because AUTO

stops giving line numbers when two < CR> 's are received.

3. Use labels after GOTO and GOSUB statements. Do not assign line numbers (except as noted in 4 below) since they will change.

4. ON GOTO and ON GOSUB cannot use labels. The trick here is to assign a line number way high in the program count. Then, at these lines, use GOTO ..label. NOLINES.BAS shows how this is done.

Use the AUTO statement to segment your program. AUTO can number starting at any location. This is useful to place ON GOTO and ON GOSUB locations.

AUTO is terminated when two sequential < CR> 's are received.

PROGRAMMING TIPS

Manuals can be full of information. Sometimes it's overwhelming. This section presents a few tips our customers have given us over the past 15 years.

Finding variables, keywords, and labels
The FIND statement will search for program labels, variable names, or even command keywords. Look in the *CAMBASIC Programming Manual* for more information.

Faster, shorter IF-THEN's
IF-THEN statements are based on zero and non-zero flags. Consider the following program fragment

```
a = 5
if a then ..there
```

executes quicker than

```
a = 5
if a< > 0 then goto ..there
```

Notice 2 elements are missing First is the inequality test < > . Next is GOTO. The program line is an implied GOTO.

Another variation is

```
a = 5
if a goto ..there
```

The THEN is implied but you must supply the keyword GOTO or GOSUB.

Use parenthesis (or brackets)
When IF-THEN statements do not execute properly (and formulas too), break it up using parens. For example:

```
IF A - B*C AND D = C*B - D/8 OR 15 THEN ...
```

The question the compiler asks is "What are you doing here?" True, there is an order of precedence, but often times the assumptions made by the programmer vs compiler order will differ. Use parens as shown below.

```
IF A-(B*(C AND D))=C*B-((D/8)OR15) THEN
```

Generally, you can go 7 levels of parens without reaching the compilers limit.

WHERE TO GO FROM HERE

If you want to do this	Go to chapter
Save a program	3
Autorun a program	3
Know more about serial ports	4
Use the SPI port	4
Using RAM to save variables	5
Storing variables in Flash	5
Configure digital I/O lines	6
Read switch status	6
Use high current outputs	6
Connect an external opto rack	6
Using calendar/clock	7
Reading voltages (Analog input)	8
Analog output	8
Using a keypad	9
Character display port	10
Sound or timer output	11
Using interrupts	13
Using high speed counter	14
Graphics display	15
Power input and output	16
Expansion port	16

Also, refer to the table of contents for a listing of major functions.

TROUBLESHOOTING

You probably turned to this section because you could not get the sign on message. The following are troubleshooting hints:

1. Check the power on the RPC-2350. A good place is C13+ and -. If it is below 4.65 volts, the RPC-2350 will be reset. Power is 5 ±0.25 volts. If it dips intermittently to 4.65 volts (due to switching noise or ripple), the card will reset for about 100 ms. If the noise is frequent enough, the card will be in permanent reset. Check U14, pin 6. If it is low (about 0 volts), then it is in reset. This line should be high (about + 5 volts).

   If you are using a 6-15V supply, make sure it does not dip below 6V even intermittently. Use a scope to make sure it does not. A voltmeter may not be good enough.

   The same can be said for the 5V supply. Make sure it does not dip below 4.8V using an oscilloscope.

2. Check the COM1 port. COM1 is also known as console port J1. Remove the connector from COM1. Refer to the outline drawing earlier in this chapter. Connect an oscilloscope (preferred) or a voltmeter to pin 3 (Txd) and ground. Pin 3 should be -6 volts or more negative. (Pin 1 is designated by the ^ symbol on the connector. Pin 3 is next to it, nearer the key opening.) If you have -6 volts or more, press the reset switch. If you have a scope attached, you should see a burst of activity. If you have a volt meter, you should see a change in voltage. Using a Fluke 8060A set to measure AC, you should see a momentary reading above 2 volts. Press reset several times to make sure it captures it.

3. Install the cable and make sure the voltages and output activity are still there. Output is from pin 3 on the VTC-9. Check to make sure something is not shorting the output.

4. Check the serial parameters on your PC or terminal. They should be set to:

   19200 baud, no parity, 8 data bits, 1 stop

If all of this fails, call technical support listed at the front of the book.

CHAPTER SYNOPSIS

- Saving programs to Flash
- Retrieving programs from Flash
- Load and run two programs
- Autorun on reset or power up
- Installing 128K or 512K Flash

DESCRIPTION

Two to 8 programs can be stored in socket U3. U3 normally has a 29C010A Flash type EPROM, which stores 2 programs. Its capacity is 128K bytes.

U3 socket can accept a 29C040A type EPROM. This type has 512K byte capacity and can store up to 8 programs.

You can store two to eight programs up to a maximum size of about 33K bytes each. A general rule to determine program storage requirements is one line requires 40 bytes. 32K bytes would store over 800 lines of code. Your application could be significantly more or less, depending upon the number of commands / line, comments, and print statements. Another indication of program size is to use the file length as saved on a PC disk.

The 29C010A (128K) Flash can store 2 programs while the 29C040A (512K) stores up to 8 programs.

Despite the fact you have 128K or 512K RAM installed, the maximum program size CAMBASIC can run is about 33K (leaving room for variable storage). Programs can be chained, however. When programs are chained, variables from one program can be used in another.

Data can be saved to and retrieved from this device using the SAVE and LOAD commands. The maximum amount of data that can be accessed is limited by the size of your program less the Flash size.

The Flash EPROM is non-volatile (retaining data even when power is disconnected), having an unlimited number of read cycles and a limited number of write cycles (about 1,000 to 10,000). Using 1,000 cycles as a limit, you can write to Flash once a day for about 3 years before “wearing out” the device.

A program is not run from EPROM. It is transferred to RAM and run from there. Programs in RAM are run and can be modified. They can be saved to EPROM for execution later.

The RPC-2350 is set to autorun on power up or reset by installing a jumper (W11). When autorun is on, the program in EPROM is loaded into RAM and begins to execute immediately.

The EPROM is write-protected with a software lock, so accidental writes on power-on or -off are almost impossible. You cannot disable the lock except when executing the SAVE command.

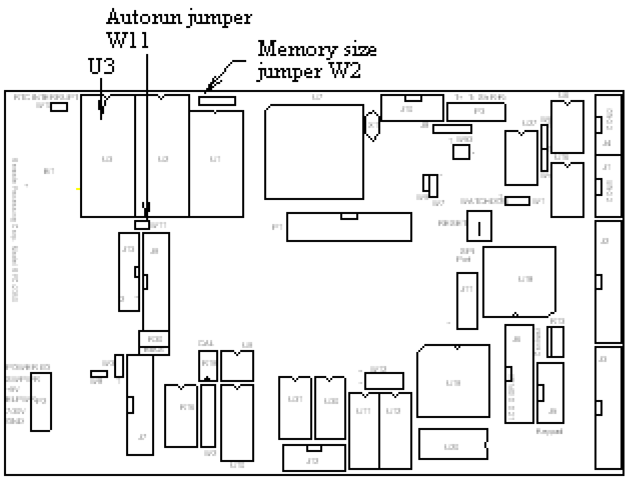


Figure 3-1 Flash location and jumpers

SAVING A PROGRAM

To save a program, set jumper W11. You may set the jumper even if the power is on. Remember to discharge any static electricity before installing or removing the jumper. For this example, assume you wanted to save the following program:

```
10 FOR N = 0 TO 2
20 PRINT "Hello ";
30 NEXT
40 PRINT
```

If this program is not already in, type it in now (or, if you prefer, use your own program).

You can save your program to any one of two locations: 0 or 1. Programs saved to location 0 will automatically run at power up. Syntaxes to save a program are:

SAVE
SAVE *program*

Where ‘*program*’ is 0 or 1 for 128K Flash or 0 to 7 for 512K Flash installed.

“SAVE” and “SAVE 0" are the same. Both save to program location 0.

Type in the following command for this example:

```
SAVE
```

CAMBASIC will compile the program, program the EPROM, and verify its contents.

```
Compile...Write...Verify
```

The time it takes to do all of this depends upon the length and complexity of the program. Generally, it will be from 1 to 20 seconds. The ready prompt (>) will appear when the program has been successfully saved to the EPROM. If the program does not write to the EPROM, an error message will appear:

```
Fail @ xxxx
```

Saving a program overwrites the previous one. There is no way to recover it since both occupy the same space.

AUTORUNNING

The program stored using SAVE or SAVE 0 will autorun. To autorun a program:

1. Make sure there is a program in EPROM (from above) stored by using SAVE or SAVE 0.

2. Make sure the autorun jumper W11 is installed.

If you push the reset button, the program should run. If there are any errors, the program will stop (assuming you have not trapped them with ON ERROR) and display the error message.

PREVENTING AUTORUN

When troubleshooting or maintaining a program, it's not always convenient for an autoexecute file to run. This is especially true if the program has been configured to ignore the <ESC> key. To prevent autorun, remove jumper W11.

Later, if you wish to SAVE or LOAD a program, reinstall this jumper. You may do so even if the power is on and a program is running. Remember to discharge any static electricity before installing or removing the jumper.

Another way to prevent autorun is to remove the signature bytes from Flash. This is done by writing &FF's into Flash address 0 and 1, segment 8. Use the following steps to do this.

```
dpoke &8000,&ffff
save 8,0,0,0,&8000,2
```

This will overwrite the first 128 to 256 bytes of code and prevent autorun.

LOADING PROGRAMS

There are times when you may wish to temporarily modify or otherwise test out a change to a program. Since the program is loaded into RAM, modifications can be made without affecting the program in EPROM. If you find out that modifications are not desirable or did not work, you can restore the original program to RAM using the LOAD command.

This version of CAMBASIC can link and run 2 programs with a 128K or 8 with 512K flash. Because of this linking ability, you should do some things during development to limit problems. Before loading a new program, enter the following command first:

```
new
```

If you are loading one program through Flash, saving it, stopping, then loading another program and running it, you may notice some oddities, like incorrect variable values or syntax errors to lines not in your code. This oddity is probably due to the program size difference. If you do not type in NEW to start with, LOADing will clobber what is in memory and nothing will be right.

If you are using multitasking routines, such as ON BIT or ON TICK, then it is best to reset the board and then load the program during development.

The other (and best) way to take care of this problem is to reset the board, then load the program.

Use LOAD 0 to retrieve programs saved using SAVE 0 or SAVE and use LOAD n to retrieve programs saved using SAVE n. ‘n’ is 0 or 1 for 128K Flash and 0 to 7 for 512K.

SAVING DATA TO FLASH EPROM

Additional data, such as strings and constants, can be saved to U3 using a variation of the SAVE command. Exactly how much memory you have depends upon how many and how large of program(s) you have.

Flash sector size must be considered when saving. The sector size determines the *minimum* write area. Thus, if you were to save just 1 byte, 128 (29C010A type) or 256 (29C040A) bytes will be written. All unwritten data is saved as FFH.

Font are stored starting at address &AC00, segment 9. This is above program area 1. If space is tight, consider using a 512K Flash.

Chapter 5 treats saving data to RAM and Flash more extensively.

INSTALLING 128K OR 512K FLASH

Socket U3 can hold a 128K or 512K Flash EPROM. Perform the following steps to install a a new Flash:

1) Remove power from board and remove existing EPROM from U3.
2) Orient the new Flash so pin 1 is near the edge of the board. Install the IC.
3) Set jumper W2 according to memory size:

W2[4-5]	128K (29C010A, 29C010)
W2[5-6]	512K (29C040A)

LINKING PROGRAMS

One CAMBASIC program can load and run another. For example, the autorun program can, at some point, run the program in Flash area 1, which can then call the original one. Using this technique, you can have program sizes of nearly 70K bytes using a 128K Flash or 245K using a 512K Flash..

This linking is intended to be called occasionally (although we have run programs that have linked millions of times). For example, the second program is a set up and calibration routine. This will free up main program RAM for variables.

Linking in CAMBASIC is not threaded code. It does replace one program with another while keeping variables.

When developing programs and testing for linking, make sure you save *any* changes to Flash before running. Any changes in program length, especially the first (autorun) one, can have adverse, strange, or disastrous results. When you have saved the changes, press the reset button to simulate an autorun. **NEVER** start with program 1 (LOAD 1) then have it load program 0.

There are other do's and don'ts, precautions and limitations to linking. They are discussed below.

Program size
The first (autorun) program run **MUST** be larger than the second, linked program. This is because variables and data are stored above the program. If the second program was longer, it would wipe them out. To simply increase program size, add comments.

Multi-tasking
All multitasking (ON BIT, ON COM$, COUNT, etc.) and interrupts are disabled and multi-tasking data cleared (hash table is zeroed out) when a program is linked. You must re-enable them in your code. The reason for this is the operating system stores the address (not line number) to execute. When you change programs, addresses change.

You can save some selected data by writing it to a variable. For example, the current count can be saved just before you LOAD 1 RUN. You will need to re-enable the count when you enter the program.

DATA statements
Use RESTORE to reset the DATA statement pointer every time you enter a program (if you are using DATA statements).

Linking within the program
To link to another program, execute

```
LOAD n RUN
```

on its own line. Do not put any other code after it. Do not make it as part of a conditional statement, as in

```
    IF A = N THEN LOAD 1 RUN
```

If you link within a subroutine, DO..UNTIL, or FOR..NEXT, execute EXIT CLEAR before LOAD.

```
.
.
2250  GOSUB 7000
```

```
.
.
7000  IF A = 0 THEN 7100
.
.
7100  EXIT CLEAR
7110  LOAD 1 RUN
```

The above example shows how to link to another program within a subroutine. If you do not use **EXIT CLEAR**, you will eventually end up with an **<Out of memory>** error.

Declare variables
All variables, dimensioned arrays, and string variables should be declared in the first program. These variables are accessible to both programs.

Program Re-entry
There is a good chance you will have declared a dimensioned array or made room for strings in the autorun program. If you do a CLEAR again, all variables will be cleared.

What you must do when you enter a program is to test for a re-entry flag at the start. If it is 0, then the program knows to go and do variable initialization. If it is not 0, then it skips around it. For example:

```
10    IF FLAG <> 0 THEN 100
20    CLEAR 500    :'Clear space
30    DIM WEIGHT(100)
40    DIM B$(20)
   .
   .  More initialization code
   .
100 FLAG = 1:'Signal initialized
110 ON ERR GOTO 10000
120 CONFIG COM$ 2,0,8,0,1
130 ON COM$ 2 GOSUB 2000
```

The code will not skip around the multitasking statements when it is first Autorun. This is because, by default, all Basic variables are reset to 0. When you link programs, variables remain intact.

SAVING PROGRAMS VS SAVING DATA

A program number and Flash segment are related by the following formula:

Flash segment = *program* number + 8

If you intend to save data to Flash, make sure you do not write to a program area. Programs are always saved starting at address 0 and continue until end of program. You can save data above a program.

The important thing here is to remember that saving programs to Flash and saving data to Flash are related.

COMMANDS

The following is a list of CAMBASIC commands used for saving and loading programs.

Command	Function
LOAD	Transfers program from U3 to RAM for editing or running.
LOAD n	Transfers program of specified area of U3 to RAM.
LOAD *data*	Multi-part syntax command that moves data from Flash to RAM or RAM to RAM. Refer to *Chapter 5* or the CAMBASIC manual for more information.
SAVE	Saves a program from RAM to U3 for Autorun
SAVE *n*	Save a program from RAM to a program area of Flash.
SAVE *data*	Multi-part syntax command that saves data from RAM to Flash. Refer to Chapter 5 or the CAMBASIC manual for more information

CHAPTER SYNOPSIS

- Overview of RPC-2350 serial capabilities
- Using RS-422/485
- Networking with RS-485
- Preventing program stops (breaks)
- SPI port information

DESCRIPTION

The RPC-2350 has two serial ports that can be used for interfacing to a printer, terminal, or other serial devices. A SPI port is also provided. This chapter describes their characteristics and how to use them. Frequent references are made to commands listed in the *CAMBASIC Programming Manual*. Please refer to this manual for more information.

Serial ports are numbered COM1 and COM2. COM1 is used for program development. It is RS-232 only. During run time, it can be used for other functions such as writing to a printer or serial display. COM2 is a general purpose serial port. Its outputs can be RS-232, RS-422, or RS-485 level compatible.

Both ports support XON/XOFF protocol to control data transmission. Each port has a 256 character interrupt driven input and output buffer. This allows characters to be sent out (using PRINT) without slowing down program execution. However, if the PRINT buffer fills, program execution is suspended until the buffer empties. Both ports have a 256 character input buffer. When more than 256 characters have been received, extra ones are ignored.

The baud rate, parity, data length, stop bits, and com port type are changed using the CONFIG BAUD command.

COM1 SERIAL PORT

COM1 is J1 and is called the Console port on the card. This port uses a VTC-9F serial cable to connect to a PC and other external serial devices. The cable is wired one-to-one, with pin 1 on the 10 pin connector lining up to pin 1 on the DB-9. The connector plugs directly into a 9 pin serial port connector on a PC.

This port is normally used for programming. During run time it may be used as a general purpose serial port. When used for programming or with the INPUT statement, it will accept ASCII character values from 0 to 127. When used with the INKEY$ and COM$ functions, it returns ASCII values from 0 to 255.

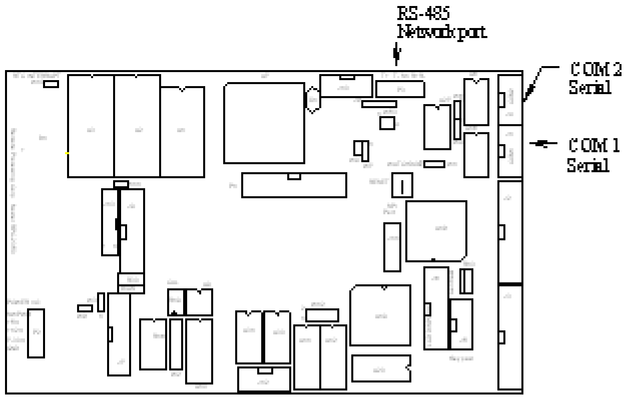


Figure 4-1 Serial port connector location

No control lines (like CTS or RTS) are available on COM1. A pull up resistor is on the CTS output should a communication program or device require a positive indication the port is alive.

COM2 SERIAL PORT

COM2 is an RS-232, RS-422, or RS-485 port. It also uses a VTC-9F serial cable to connect to PC's and other serial devices. COM2 is identical to COM1 except that it has 2 hardware handshaking lines, CTS and RTS.

NOTE: The CTS output line is low on power up or reset. This is to prevent the RS-485 transmitter from turning on and adversely affecting network communication. RS-232 communication may be affected by holding off the sender from sending data. You must enable the CTS output as described below under "Communication flow control".

Jumper W4 determines if COM2 receive is RS-232 or RS-422/485.

W4[1-2]	RS-422/485
W4[2-3]	RS-232

Use CONFIG BAUD to tell the system the type and characteristics of this port. Default is RS-232, 19,200 baud, 8 data, 1 stop bit..

Communication flow control
There are two methods to control transmission: Hardware and software. Both are discussed below.

When RTS line in goes low, the RPC-2350 is held off from transmitting out COM2. The status of this port is read by the BIT statement. The example below returns the status of the RTS line:

```
100 B = BIT(130,5)
```

If B = 1, transmission is held off.

You should check this line before executing a PRINT #2 statement. If transmission is held off, the print buffer can become full. PRINT #2 command will 'hang' until the buffer accepts all characters in the program line. This may give the appearance of the program locking up or running very slowly. This could also result in a "deadly embrace" situation where the receiving device is expecting some other condition before it will receive characters.

The CTS line may be set high or low to control communication from a device when using RS-232. This line is also used in RS-422/485 communication at P3 or J3 to control transmitter output.

When using RS-232 communications, you may have to initialize the CTS output to high to allow the sender to transmit. The examples below show how this is done. Line 400 sets CTS low and 500 sets it high.

```
400 BIT 128,4,1
500 BIT 128,4,0
```

A low condition is supposed to hold off a device. Some devices do not recognize CTS output so this may be a mute point.

CTS is used for RS-422/485 communications to turn on/off the transmitter. Use CONFIG BAUD to not only set the baud rate but also the type of communication protocol. CAMBASIC takes care of the CTS line for RS-422 and RS-485.

The CONFIG BAUD statement sets both serial ports for baud rate and type (RS-232, RS-422, and RS-485) for COM2.

XON/XOFF
The so-called XON/XOFF protocol is a software scheme to enable and disable transmitting. Sending an XOFF character (ASCII value 13H or < Ctrl-S>) to the RPC-2350 will stop transmission until an XON (ASCII value 11H or < Ctrl-Q>) is received. This does not over-ride the RTS input line when RTS is low.

RS-232
The most common communication interface is RS-232. Quite simply, RS-232 defines only '0' and '1' voltage levels and transient speeds. It does not define baud rate, start and stop bit length, parity or the protocol. RS-232 is available at J1(COM1) and J4 (COM2).

Usually, when people speak of RS-232, they mean something that is compatible with a PC. However, you must make sure the baud rate, parity, and stop bits are proper for your device. These are set in the CONFIG BAUD command.

RS-232 is used in point-to-point communication. That is, one device talks to one other device. Schemes have been devised to network RS-232.

The maximum baud rate RS-232 supports depends upon the distance and cable characteristics. As a rule of thumb, 9600 baud will work reliably to 50 feet.

RS-422
RS-422 is used for long distance (1000 meters) communication. It is similar to RS-232 in that it is point-to-point and the communication "protocol" is the same.

The RS-485 port is used for RS-422 communication. The transmitter is always on. If, for some reason, you want to shut the transmitter off, execute

```
BIT 128,4,0
```

in your code. To turn the transmitter back on, execute

```
BIT 128,4,1
```

in your code.

RS-485
RS-485 is a popular networking system. Technically, RS-485 only defines electrical specifications, not protocols. (See Multi-drop Network below.) In RS-485, the transmitter is turned on only long enough to send a message, then it turns off. Thus, RS-485 allows multiple units to be put on a single set of wires.

RS-485 operates in one of two modes: 2 wire and 4 wire. (An extra wire is actually needed as signal reference.) There are pro's and con's to 2-wire vs 4-wire systems and they are not discussed here.

Mechanically, to make a 2-wire system, simply connect T+ to R+ and T- to R- on P3.

When you connect the RPC-2350 to another device, the T+ ,T-, R+ and R- signal lines may be reversed. That is, you may need to connect T+ to T- and so on. This is due to naming convention confusion when RS-485 was first introduced.

RS-422/485 OPERATING INFORMATION

RS-422/485 Termination network
When the RPC-2350 is the last physical unit on a network (RS-485), or it is the only unit (RS-422), the receiver must be terminated to prevent ringing. Insert jumpers in W5[1-2][3-4] to install the network terminator. See figure 4-2 below.

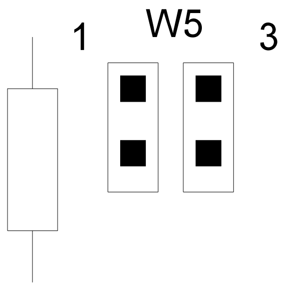


Figure 4-2

Only one slave device on a RS-485 network should have a terminator installed. The host transmitter should also have a 100 ohm resistor in series with a 0.1 mfd capacitor across its T+ and T- terminals. The terminator on the RPC-2350 includes pull up and pull down resistors to prevent lines from floating and generating erroneous characters.

Two- and four- wire RS-485
The RS-485 port on the RPC-2350 is set up for 2- or 4-wire mode through jumper W6. This jumper either always enables receive (W6[2-3]) or enables it only when the transmitter is off(W6[1-2]).

W6[1-2]	Receive off while transmitting (2-wire)
W6[2-3]	Receive always on (4-wire mode)

RS-485 Transmitter turn-off
The RS-485 transmitter is automatically turned on and off by CAMBASIC operating system. You must specify RS-485 mode in the CONFIG BAUD 2 command.

During two wire mode and W6[1-2] is set, the receiver is turned off when the transmitter is turned on. This keeps the RPC-2350 from receiving what was just sent out.

NOTE: Do not operate RS-485 at 150 baud. Automatic turn off is not adequate. Contact RPC for suggestions.

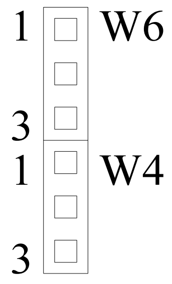


Figure 4-3 Jumper W4 & W6 detail

Two-wire RS-485
Mechanically, to make a 2- wire system, simply connect T+ to R+ and T- to R-. Make sure CONFIG BAUD is set up for RS-485 mode. Set jumper W6[1-2]. This prevents data transmitted from getting received.

Network response time considerations
When the last character is sent out, an internal timer is activated. When the timer is done, the RS-485 transmitter is turned off. The timer shuts off the transmitter about 1/4 to ½ character time after the last character is sent out. The difference is due to stop bit and parity requirements and to allow for line settling.

Normally, this is not a problem. However, if a very high speed system is controlling the network, it is possible two transmitters can be on simultaneously, garbling data. Any responding systems on the network should wait at least ½ character time before sending a message packet. When using other 2350's on a network, this will not be a problem at 19200 baud and probably will not be one at lower rates.

Multi-drop Network
You can use the RPC-2350 in a multi-drop network by using COM2's RS-422/485 port. You can connect up to at least 32 units (including other RPC-2350's) over a 4,000 foot range.

The host transmits data packets to all of the devices, or nodes, in the network. A data packet includes an address, command, data, and a checksum. See figure 4-4. The packet is received by all devices, and ignored by all except the one addressed.

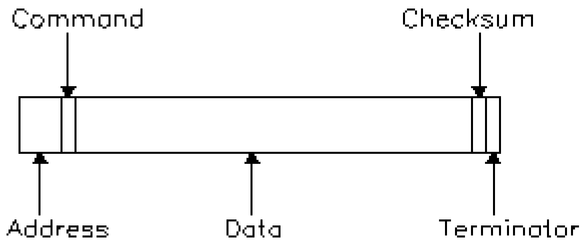


Figure 4-4 Data packet

The relationship described below between nodes and the host is a master-slave. The host directs all communication. Nodes "do not speak unless spoken to". Peer to peer communication, while possible with the RPC-2350, is not discussed here.

There are many communication protocols. For this example, a protocol might look something like this:

> 22MB1

The protocol starts with the < cr> character. This character synchronizes all units and alerts them that the next few characters coming down are address and data. In this case, "> 22" is the units address. "M" is the command and "B1" is the checksum. The command is terminated with a < cr> character.

Response depends upon the nature of the command. Suppose the command M means "return a digital I/O port status". The RPC-2350 could read the port and respond with AA2< cr> . The first A is an acknowledge, that is no errors were detected in the message. The data, A2, can be broken down as follows:

Bit/line	7 6 5 4 3 2 1 0
Status	1 0 1 0 0 0 1 0 = A2

Lines 1, 5 and 7 are high while the others are low.

The following program fragment uses ON COM$ in a network environment. ON COM$ generates an interrupt when a < CR> is received. The interrupt program uses INSTR function to determine if the data packet was addressed to this card.

```
10 CONFIG BAUD 2,7,4,0,2
20 CONFIG COM$ 2,13,0,0,1
30 ON COM$ 2 GOSUB 1000
40 UNIT$ = ">05"
   .
   .
   .

1000 PACKET$ = COM$(2)
1010 A = INSTR(0,PACKET$,UNIT$)
1020 IF A = 0 THEN RETURN
   .
   .
```

Line 20 sets up ON COM$ to interrupt on a < CR> and branch to line 1000. Line 40 sets up this card' s address.

Line 1010 checks to see if the received message = this card's address. If not, the subroutine ends. When there is a match, further processing is performed.

An application program, 485TST.BAS, filters out control codes (such as < LF> and < CR>) at the start of the message. Since CONFIG COM$ set up a communication interrupt on a < CR> , your sending device can also send a < LF> .

485TST.BAS also checks for communication errors. Its main purpose is to detect communication errors but it also acts as a good foundation for a communication program.

485NET.BAS is a simple networking program. To test it out, jumper W4[2-3] temporarily. After the program is running, connect your terminal to J4. Instructions are printed when the program is first run.

ACCESSING SERIAL BUFFERS

You can access COM1 and COM2 buffers in three ways:

1. INPUT statement. This removes all characters in the buffer up to the terminator character and puts them into a variable.

   When using the INPUT statement, program execution is suspended until a < cr> (Enter key) is received. Whether this is a problem depends on your particular application.

   INPUT strips bit 7 on the COM1 port. This means ASCII characters from 0 to 127 are received. The INPUT statement can return a maximum string length of about 150 characters.

2. INKEY$(n) function. Characters are removed one at a time. A null string is returned when the buffer

is empty.

In this mode, you have access to the full 256 bytes. If you don't read the buffer and the buffer fills, all subsequent characters are discarded. INKEY$(n) may be used anywhere in the program.

3. COM$(n) retrieves all characters in the buffer, including < cr> 's and other control codes. This function is commonly used with ON COM$ multitasking statement. You can retrieve 128 of the 256 bytes in the serial buffer at one time.

DISABLING PROGRAM BREAK

Program execution can be interrupted by pressing the < Esc> key. To disable this recognition so the program does not terminate, put the following command in your program:

```
1000 CONFIG BREAK port,1
```

Where **port** is 1 (J1) or 2 (J4 or P3).

SPI PORT

A software SPI port is provided at J11. SPI (Serial Peripheral Interface) is used to communicate with a number of IC's. These include D/A's, A/D's, UART's, and other devices. J11 provides two chip selects.

The optional touch screen interface uses SPI port 1.

The SPI function is used to read and write data. Unfortunately, SPI has a variety of data formats. Data to send and receive from a device can be anywhere from 8 to 24 bits. The clock polarity and data phase can idle high or low when data is latched.

CAMBASIC SPI function supports the following format:

Clock idle polarity: low
Clock-data phase: low

This format supports the Maxim MAX186/188 and Burr-Brown ADS7843 IC's. The program SPIDEMO1.BAS uses the MAX186/188 to return the result of an A/D calculation. Functionally, it is the same as the AIN command.

If your format needs are different, there is a CAMBASIC program (SPIDEMO2.BAS) that you can use as a basis to read and write to other SPI devices.

Command format is:

a = SPI(*channel,out_length,data, delay,in_length)*

Where:

channel = 0 to 2, the SPI channel number.

out_length = 0 to 16, data output length in bits. When zero, no data is shifted out. *data* can be any value but must be included.

data = 0 to 65,535, command/data to send to SPI device.

delay = time to wait before retrieving information from SPI port. Time in micro-seconds is calculated as follows: time = *delay* * 1.1 + 4. If 0, there is no delay. Use 0 if there is no data to retrieve (i.e. sending to D/A).

in_length = 0 to 16, data input length in bits. Will return a number from 0 to 65535.

The table below shows the location of SPI device selects. A device is selected when a line goes low.

SPI port number	Location
0	Analog input (U10)
1	J11-4
2	J11-5

The table below is the pin out for SPI port J11.

Pin No.	Description
1	Clock output to device
2	Serial data to external device
3	Serial data from external device
4	SPI 1 select (active low). Used for touch screen select.
5	SPI 2 select (active low)
6	+ 5V supply
7	Ground

SPI Port connector type information
The SPI port connector header is a Molex-Waldom type. Its part number is 22-23-2071. This is a 0.1" center, 0.025 post connector.

The mating connector (terminal housing) part number is 22-01-3077. Crimp terminal part numbers are 08-50-0114 (tin plate) or 08-55-0102 (gold). A low cost crimping tool is 63811-1000. Parts and tools are available from Digi-Key (800 344 4539 or www.digikey.com)

Other housings and terminals are available (such as high pressure). Refer to a Molex data book for more information. AMP MT connectors (such as 87499-1) can also be used. They fit, but are not polarized.

SERIAL PORT FILE NUMBERS

CAMBASIC references the serial I/O ports by file numbers. The following table lists the corresponding file number to serial I/O port and how they are used with the various ports.

Description	File	Examples
COM1	1	PRINT "Hello" PRINT #1,"Hello" INPUT A\$ A\$ = INKEY\$(1)
COM2	2	PRINT #2,"Hello" INPUT #2,A\$ A\$ = INKEY\$(2)

COM1 is J1, the console port. COM2 is J4, the primary port.

COMMANDS

The following is a list of CAMBASIC commands used for serial I/O. Some variations are not listed here. Refer to the *CAMBASIC Programming Manual* for more information.

Command	Function
CLEAR COM\$	Clears serial input buffer
COM\$	Returns string from buffer
CONFIG BREAK	Prevents < ESC> from stopping the program
CONFIG BAUD	Sets serial port parameters
CONFIG COM\$	Configures port for ON COM\$(n) interrupt
INKEY\$	Returns a character from the serial buffer
INPUT	Receives string or number from port
LIST	Outputs program listing
ON COM\$	Calls subroutine on serial input
PRINT	Outputs data in various formats
SPI	Serial I/O for external IC's
TAB	Tabs to predetermined positions

SERIAL CABLE PIN OUT

The following is the pin out between the IDC connector for the RPC-2350 and the DB-9 connector to the PC or terminal.

IDC pin #	DB-9 pin #	Description	Direction from 2350
1		nc	
2		nc	
3	2	Transmit	Out
4	7	RTS	In
5	3	Receive	In
6	8	CTS	Out
7	nc		
8	nc		
9	5	Ground	
10	na	+ 5V	Out

Note that CTS and RTS are not available on J1.

The VTC-9 serial cable is a simple one-to-one connection between a 10-pin IDC connector and 9-pin IDC DSUB.

CHAPTER SYNOPSIS

- Battery backup description
- Where and how to store variables
- Saving and retrieving data from Flash
- Installing 512K RAM
- Discussion about corrupted data

DESCRIPTION

The RPC-2350 is usually installed with 128K of RAM in socket U2. An optional 512K can be installed. RAM is battery backed on both models.

This chapter discusses saving and retrieving variables from RAM and Flash EPROM and running assembly language programs.

If program and data are battery backed, the UNNEW command may be used to try to restore the program. Variables used by the Basic program are cleared, however. Certain variables are preserved and data POKEd into RAM is saved.

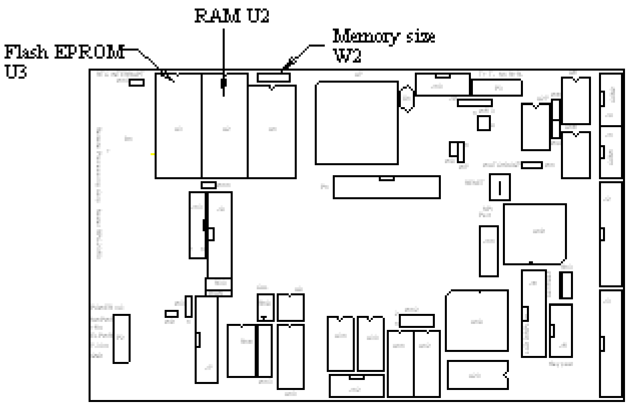


Figure 5-1 Memory location

BATTERY BACKUP

Battery life is about 3 years (27,000 hours). See additional information a few paragraphs below.

Battery B1 is used to back up the RAM and real time clock

The installed battery is rated for 190 mA-hours. RAM (any size) *typically* requires 1-2 micro-amps in standby. The clock requires *typically* 3 micro-amps. Assuming 7 micro-Amps of current draw, you could expect 3 years life from the battery (assuming the board was off the whole time.)

The problem with calculating battery life are variables beyond our reasonable control. First, memory manufacturers specify a ‘typical’ current of 1 to 2 micro-amps and a maximum of 100 (high temperature operation). Other factors affecting battery life include operating temperature, clock chip, and time the RPC-2350 has power applied to it. You can expect the battery to last between 3 to 5 years for operation at 25°C. At 50°C, life is about half as much. This is due to battery deterioration and CMOS leakage increases at higher temperatures.

Humidity also affects battery life. Very high humidity (in conjunction with a dirty environment) increases leakage. Low humidity drys out the battery seal, allowing contaminates to enter.

The point of this explanation is to give you the factors affecting battery life. Under the best of conditions, life is 7 years. Under worse conditions, it could be as low as 3 months. You can add a larger battery as described below.

Existing battery voltage is measured across W14.

Alternate battery
A larger 3.0V battery can be installed by connecting it to W14. Be sure to remove the existing battery. Note the polarity marked on the board.

STORING VARIABLES IN RAM

The term "variables" in this context includes numbers, strings, arrays, recipes, and formulas as applied to your application.

CAMBASIC provides 26 “protected” variables, A%-Z%, that are not erased on power up. These are accessed like other BASIC variables.

The Flash EPROM may be used to store variables or constants, such as text strings, to help reduce the size of the main program.

CAMBASIC memory map
The following diagram is a memory map for the RPC-2350.

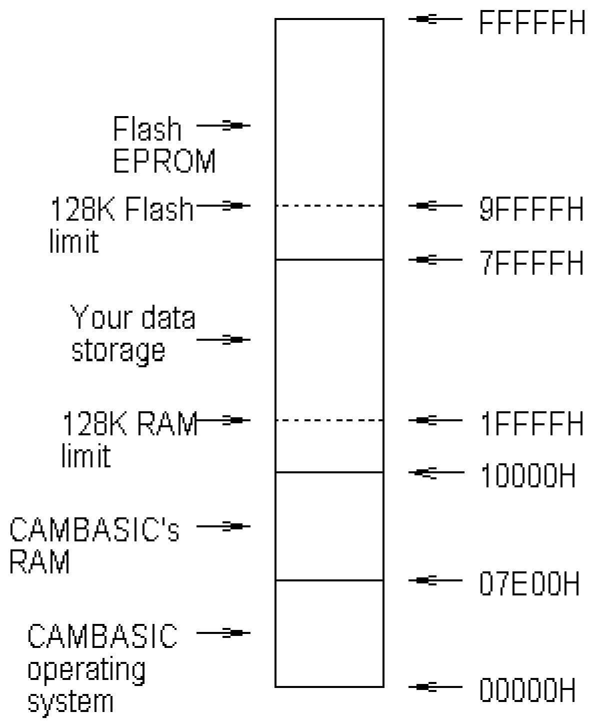


Figure 5-2 2350 system memory map

Programs and CAMBASIC variables reside in segment 0, between address 00000H and 0FFFFH. Your variables (as defined above) are stored from 10000H to 1FFFFH (with 128K RAM), which is called segment 1, address 0000H to FFFFH. When 512K RAM is installed, the useful range is from 10000H to 7FFFFH. A segment has an address range from 0000 to FFFFH (or &0000 to &FFFF using CAMBASIC notation) (in decimal terms, this is 0 to 65535).

NOTE: Do not use the CAMBASIC SOUND statement when the board has 512K of RAM or Flash memory. Sound output is multiplexed with an address line.

Program and basic variables (A, B(15), C$, etc.) always reside in segment 0 and are cleared on reset. A special set of variables, A% - Z%, reside is segment 0 and are not cleared on reset. These are floating point numbers and can be used like any other variable in Basic.

Variables you peek and poke to should reside in segment 1 with 128K RAM installed or segments 1-7 with 512K. PEEK and POKE commands store and retrieve values from memory. For example:

```
20 POKE 12,A,1
```

puts the value of A into segment 1, address 12. Use the PEEK statement to retrieve the variable:

```
50 B = PEEK(12,1)
```

You can store and retrieve arrays, strings, and variables in this way. There are many variations of PEEK and POKE statements. Refer to the *CAMBASIC Programming Manual* for additional information and examples. A list of commands appears at the end of this chapter.

Flash Memory

Programs are stored in Flash EPROM. Programs are transferred from Flash to RAM at run time or LOAD. Data may also be stored in Flash. Below shows the Flash memory map.

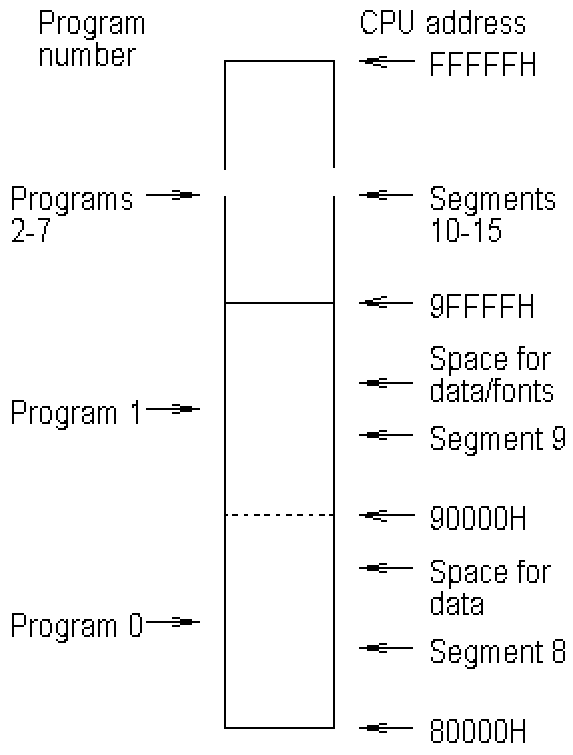


Figure 5-3 Flash memory map

A 128K or 512K Flash type EPROM may be installed in U3. Jumper W2 configures U3 for Flash size.

Data may be stored above the program in Flash using the SAVE command. The SAVE command transfers data from RAM to Flash. Fonts for the RPC-2350G are stored in Flash above program 1 starting at Flash address &1AC00 (this is segment 9, address &AC00 when using SAVE). This will not interfere with your program but will affect any data storage plans you may have.

A CAMBASIC program number and Flash segment when using SAVE are related by the following formula:

SAVE segment = program + 8

Keep track of where you are writing to. Most programs will only use program area 0. Addresses above &9000 are always available in any program area. However, if you have many programs, you will have to keep track of where you are saving data to make sure a program does not get clobbered.

LOAD is used to transfer data from Flash to RAM. An example of data include arrays used in CAMBASIC. This is explained further below.

Saving and Initializing Arrays and data

Sometimes it is convenient to save an array of information to battery backed RAM and/or Flash for retrieval later. This is a handy way of storing "recipes" or tables of information for each job, customer, or process.

Arrays are initialized and filled by the basic program. Then, they are saved to either RAM or Flash EPROM. The number and size of arrays that can be saved is limited only by available memory. The saved arrays can be retrieved at any time.

How arrays are saved and retrieved depends upon what kind of memory you are saving to and loading from. If arrays are saved to Flash, use SAVE. Use LOAD to both save to and retrieve from RAM.

An example of saving arrays to RAM is shown in ARRAY1.BAS program, on the application disk. Saving arrays to Flash is shown in ARRAY2.BAS. FLASH.BAS shows strings, bytes, and word saves to Flash. ARRAY3.BAS is similar to ARRAY1.BAS except it uses LOAD to transfer arrays. This makes transfers faster.

Mapping your stored data

A frequent question is "Where do I store my data?" and "How much do I have?" You have two places to store data: RAM or Flash. How much you have depends upon several factors:

Expected program size
How much RAM and/or Flash memory is free
Will you need a second program?
Amount of data to store
Type of data to store
How simple do you want to keep the program
How secure does the data have to be (where to store data - Flash or RAM)
How often is information updated

Every application has a different set of priorities. Some programs are large, but only a few variables are stored. Others, some data is critical and some are not. Start off by determining what your storage requirements are (that is, floating point numbers, integers, strings, screen graphics...).

64K to 448K bytes of RAM are available, depending on how U2 is populated. Each data element type requires a different number of bytes. Use the following table to determine your storage requirements:

Type	Bytes of storage	BASIC Commands
Byte	1	POKE & PEEK
Word	2	DPOKE & DPEEK
Float	4	FPOKE & FPEEK
String	1+ maximum string length	POKE$ & PEEK$
Graphic	32 - 9600 bytes	DISPLAY LOAD
Program Size	1-32K	SYS(0)

The mathematics for keeping track of addresses can get quite messy. It all depends upon your data structure. If you are trying emulate a structure in C or vectors in JAVA, you will have some math to access the right data.

The demonstration program LOGGER.BAS logs at over 2000 points using the structure below.

Suppose you are logging a process and need to store the following types of information periodically:

Name	Type	Bytes
Date	String	9
Time	String	9
Temperature	Float	4
Tick time	Float	4
Level	word	2

First add up the total number of bytes needed. For this structure it is 28.

Next, assign variable names to the offsets in memory data begins.

Name	Pointing to	Value
DSET	Date string	0
TSET	Time string	9
TPSET	Temperature	18
PSET	Tick time	22
LSET	Level	26

This is simply done by making a variable equal to a number. For example,

```
PSET = 22
```

NOTE: The word "SET" does not have any significance other than naming the variable.

You will need a pointer to track the number of data sets (structures) saved. You will also have to check this pointer to make sure you are not exceeding the maximum amount of memory. Use one of the protected variables (A%-Z%) as the pointer. This way, if power disappears, the pointer is still in tact. Protected variables can also be used to keep track of the segment used to store data in.

A sample CAMBASIC line to store data is:

```
FPOKE A%*28TPSET,value,B%
```

This example uses A% as the pointer and B% to track the segment number. 'TPSET' and B% could just as easily be constants.

This structure will save 2340 elements in 64K of RAM. Therefore, you should test A% for a limit of 2340 once each loop.

A 512K RAM can store up to 7 segments. Therefore, B% is checked for a value of 8 or more at the end of this loop.

If your data requirements are more than available RAM, you can store some in Flash. You will have to first write the data to RAM first, then save to flash.

Flash is also used to save critical information. However, Flash will "wear out" after 1,000 to 10,000 writes.

You will probably have to make an initial guess at your program size. As a practical fact, no more than about 34K of program can run at one time. This leaves about 30K in the first Flash segment and at least this amount in the 2nd. Use the SAVE command to transfer data to Flash.

You can start saving at address &8400 and not interfere with the first program. If you SAVE to segment 9 and are using graphics on the RPC-2350G, larger fonts are stored starting at address &AC00. Put a limit check at this address. When a 512K Flash is installed, segments 2-7 are available, if there are no other programs in them.

The SYS(0) function returns your program size. You can use this figure to determine where you can start saving in Flash. Be sure to round up to the next page boundary (last two bytes of the address) to &00 when determining your data start of address. This is to account for the Flash block size.

All cases limit maximum address to &FFFF in any one segment. Be sure to read "Considerations for saving to Flash" below for more information.

Since each situation is unique, call Remote Processing technical support at 303 690 1588 to discuss your problem further.

Considerations for saving to Flash

The RPC-2350G uses &5A00 bytes in the Flash EPROM to store graphics fonts. The fonts are stored starting in segment 9, address &AC00. This is high enough in memory so no CAMBASIC program will interfere with it. Only if you use medium and large size fonts on the RPC-2350G will you have to consider this as an upper memory limit for storing data. Consider installing a 512K byte flash and saving to segments 2-7.

Flash EPROM is written to in blocks, or sectors of 128 (29C010A installed) or 256 (29C040A installed) bytes each. This means if you want to save just 1 byte, 128 or 256 bytes are used. You must pay attention to sector size for two reasons. First, a sector is the minimum number of bytes written. If a program requires only 35 bytes to be saved, a full sector is written. If you had the following in your code:

```
1000 SAVE 1,5,1,&1000,35
.
.
.
2000 SAVE 1,42,1,&1025,35
```

several things happen. Data saved at line 1000 is overwritten by the data in line 2000, even though different write addresses were specified. This brings us to the second reason sector size is important.

CAMBASIC forces the requested Flash address down to an even sector address. In both cases above, data is written to the Flash starting at address 0, not at 5 or 42.

The easiest way to make an even sector address is to "AND" the Flash address with &FF80 (as is done in ARRAY2.BAS program example) when using a 128K flash or &FF00 with a 512K.

Another consideration is the number of times Flash can be written to. Atmel specifies anywhere from 1000 to 10,000 or more writes. Compared to RAM, this is quite limiting. Flash should be used to store default constants or data that changes only occasionally.

Writing takes about 90 ms/1000 bytes. During SAVE time, interrupts (ON COM$, ON KEYPAD, ON BIT, etc..) are recognized but not serviced. If these commands must be serviced quicker, write data in smaller blocks.

Using LOAD to transfer data
The LOAD command can be used to transfer data from Flash to RAM or RAM to RAM. Use SAVE to transfer from RAM to Flash.

LOAD transfers up to 64K blocks of memory at a time. Use it to transfer an entire data structure containing recipes, formulas, constants, etc.

The sample program ARRAY3.BAS shows how to move data from extended memory RAM into CAMBASIC and back.

INSTALLING 512K RAM

A 512K RAM (part number 1039) can be installed in U2. The additional RAM allows you to increase data storage, not program size.

Follow these steps to install RAM.

1. Turn off power to the board.
2. Remove existing IC from U2.
3. Install the 512K RAM. Make sure pin 1 is oriented towards the board edge. Pin 1 will be marked with a dot or notch on top.
4. Move jumper W2[1-2] to [2-3].

You are now ready to power up the board. You can now PEEK and POKE data into segments 1-7.

CORRUPTED VARIABLES

When your application must rely on the accuracy of data after power up, corrupted variables become a possibility.

The nature of RAM is it is easily written to. Any POKE'd data is susceptible to corruption. This is especially true when the board is powered down. U26 monitors the supply voltage and turns off writing when it is below about 4.65 volts. However, when POKEing long data, such as strings and floating point numbers, or writing to Flash, a power down could interrupt a saving process. The result is information is corrupted. A scenario is explained below.

A program is running and saving data. During this time a reset occurs. A reset can occur due to power loss, someone pushing the reset button, or a watchdog timer time out. The data is corrupted because the complete value was not saved.

Since it is impossible to predict or delay a reset, a work around is to duplicate or triplicate values. That is, you would have to save the same information in two or three different places. Usually you only need to save the pointers to data structures.

When you are writing arrays of data (such as shown in LOGGER.BAS), the sequence your program should follow is this: Write data to RAM. Update the pointer. This pointer could be in duplicate or triplicate. This way, you only loose one set of data.

When you are saving only one set of data, the following applies.

For purposes of discussion, data variables are called sets because it can consist of a mixture of variables, strings and arrays.

On power up, your program would compare values from one set to the other one or two. If the two (or three) agreed, then there was no corruption and the program can reliably use the values. In practice, you would read information from set 1, but would save data to all two or three.

The use of duplicate or triplicate sets depends upon what the system must or can do if data is corrupted. When using a duplicate set, a corrupted set indicates that default values (from the program) should be used, since it is uncertain if the first or second set is corrupted. Both data sets would then be re-initialized.

A triplicate set is used to recover the last set or indicate that the data in the first set is valid. The procedure and logic is as follows.

Data is written to each element in a set in a specific and consistent order (data to an entire set does not have to be written to, just that element). For example, a calibration constant is saved (POKE'd) in three different places. Assume that the constant was assigned address 0, 100, and 200 in segment 1. The data is POKEd to address 0 first, then 100, then 200.

Upon reset, the calibration value is checked. If the value at address 0 agrees with address 100 and 200, then no corruption occurred. When address 0 and 100 agree but not 200, then this indicates that a reset occurred while updating the third set. The first data set can be trusted. The third data set simply needs to be updated.

When the first two sets do not agree, then you know that the first data is corrupted. If the second and third set agree, then, depending upon the system requirements, the first set could be "corrected" using the old data. The user or other device could be alerted that a calibration (or other) must be performed again. When all three sets disagree, then you must take action appropriate to the situation.

Another technique to check for valid memory is checksums. Simply write a program to add the values in RAM and compare it against a number is a good check. However, you cannot tell which data element was corrupted.

Instances of data corruption are rare. They do increase as the board power is cycled or reset.

ASSEMBLY LANGUAGE INTERFACE

Assembly language programs (including compiled C) must start from segment 0. Use the CAMBASIC CALL statement to execute an assembly language program.

A specific area of RAM should be reserved for the program. This is to prevent strings and variables from corrupting that area of RAM. Use the SYS(1) and SYS(2) statements to do this. SYS(1) returns the lowest memory location while SYS(2) returns the upper location. Run the program first to make sure variable memory has been allocated before running these SYS commands. Failure to do so may result in address returned that are not really free for assembly language programs.

There are several ways to put a program in memory, depending upon your application.

1. Use DATA statements and POKE the code into segment 0 RAM.

2. Write a program to download code. Some applications are connected to a larger system which "initializes" its systems. Using INKEY$ or COM$, code is received and then poked into memory using POKE$.

3. Read the code from the EPROM (U3) (using INP) and transfer it to RAM (using POKE).

4. Some space is available in the CAMBASIC ROM. Space from about 6B00H to 6FFFH is available in version 1.4 of the 2350G board. The starting address will probably change in the future with different CAMBASIC versions. You may burn your assembly language program in U1 and CALL in from BASIC.

5. Space is available in the Flash EPROM. In theory you can run directly from Flash. This involves running in sectored areas unique to the Z180. However, this is probably more effort. Use the Flash to store the program and then transfer it to RAM segment 0.

In all cases, it is best to load code into RAM from a "secure" source, such as Flash EPROM. Even though RAM is battery backed, over time there is the possibility it could be corrupted.

Below is an example of loading and running an assembly language program.

```
100 FOR N = &FB00 TO &FB0C
110 READ A
120 POKE N,A
130 NEXT
900 DATA &DB, 2, &47, &E6, &FE, &D3
910 DATA 2, &78, &F6, 1, &D3, 2, &C9

2000 CALL &FB00
```

Lines 100 to 130 load the program into RAM. DATA statements may be entered manually.

Line 2000 calls the program listed below. It toggles J2 line 13.

```
IN      A,(2)
LD      B,A
AND     0FEH
OUT     (2),A
LD      A,B
OR      1
OUT     (2),A
RET
```

EXAMPLE PROGRAMS

The following is a list of CAMBASIC programs used to save and load data to and from RAM and Flash.

Name	Function
ARRAY1.BAS	Moves floating point array data around RAM.
ARRAY2.BAS	Reads and writes array data to and from RAM and Flash
FLASH.BAS	Writes to and reads data from Flash
LOGGER.BAS	Data logs to RAM, prints out results.

COMMANDS

The following is a list of CAMBASIC commands used with RAM.

Command	Function
CALL	Calls an assembly language routine
CLEAR	Clears and allocates string space
PEEK	Returns a byte
DPEEK	Returns a 16 bit value
PEEK$	Returns a string
FPEEK	Returns a floating point number
POKE	Stores a byte
DPOKE	Stores a 16 bit value
POKE$	Stores a string
FPOKE	Stores a floating point number
LOAD	Move data from Flash to RAM or RAM to RAM
SAVE	Save data to Flash from RAM

CHAPTER SYNOPSIS

- ● Overview of the digital lines
- ● How to program
- ● Using high current port
- ● Interfacing to opto racks

DESCRIPTION

Digital I/O lines are used to interface with opto-module racks, switches, low current LED's, and other TTL devices. The RPC-2350 has 48 of these lines available through J2 and J3.

J3 is shared with other connectors and functions. Eight lines are high current outputs, capable of sinking 75 to 200 ma. Another 8 lines on J3 are shared by the keypad connector, J5. Still another 8 lines are used by the LCD character port J6. A table at the end of this chapter lists line use at J3.

Eight, 16, or 24 position opto racks are connected to J2 or J3. These opto racks accept G4 series opto modules. G4 series opto modules are used to sense the presence of AC or DC voltages or switch them. Maximum switching current is 3 amperes.

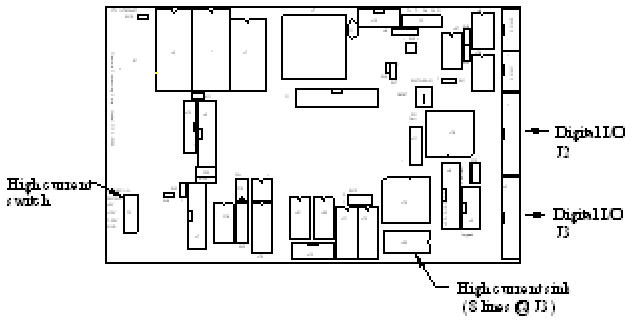


Figure 6-1 Digital I/O connectors

WARNING:

Apply power to the RPC-2350 before applying a voltage to the digital I/O lines to prevent current from flowing in and damaging devices. If you cannot apply power to the RPC-2350 first, contact technical support for suggestions appropriate to your application.

A high voltage (±15 volts) input is available at J10-10. This input is intended for the counter. However, it can be used as a digital input. Connect jumper block W9-2 to a digital input at J2 or J3.

On power-up or software reset (or CAMBASIC CALL 0), all digital ports are reset to inputs.

DIGITAL I/O PORT

Digital I/O lines on the RPC-2350 are supplied by an 82C55 chip. The chip's lines primarily go to connectors J2 and J3. Lines to J3 also go to J5 and J6. This part assumes you will be using all lines at J3 for digital I/O.

The lines on J2 and J3 are divided into 3 eight bit groups. Ports A and B can be configured as all inputs or outputs. Port C can be programmed as one group of 8 inputs or outputs or as two groups of four lines (upper and lower C). The four lines in upper and lower C can each be programmed as all inputs or outputs. Configuration is done in CAMBASIC using CONFIG PIO command.

When a line is configured as an output, it can sink a maximum of 2.5 mA. at 0.4V and can source over 2.5 mA.at 2.4V. Outputs sink 15 mA.at 1.0V.

J2 and J3 are accessed using CAMBASIC LINE, OPTO, INP, and OUT statements. LINE reads or writes to a port based on the connector number. LINE is generally used with the STB-26 board. OPTO reads or writes to an opto module based on its position in an MPS opto rack. INP and OUT access a byte of data at a port. Refer to the tables at the end of this chapter for pin outs, OPTO, and LINE references.

The base I/O address for J2 is 0 and J3 is 64 when using INP, OUT, and CONFIG PIO statements. CONFIG PIO statement is used to configure the 8255 lines for inputs and outputs. Upon reset, watchdog time out, or a CAMBASIC CALL 0, lines are configured for inputs.

J2 and J3 are accessed using LINE or OPTO statements according to the table below.

Connector designation	LINE # terminal	OPTO rack position
J2	1-25	0-23
J3	101 - 125	100 - 123

LINE #'s access the corresponding pin number on J2 or J3. LINE # 2 or 102 are not valid. This is a + 5 volt supply.

J3, port A is shared with the LCD character display port J6. If you are using J6, then these 8 lines at J3 are not available.

J3, port B is connected to a high current sink through U20. See *High current output* later in this chapter. Two lines are connected to the keypad port. They are active only when scanning a 24 position keypad.

J3 port C is shared with the keypad port J5. If you are using a keypad through J5, these 8 lines are not available at J3.

Pull up resistors
Digital I/O lines at J2 and J3 are pulled up to + 5 volts through a 10K resistor pack.

These pull ups makes interfacing to switches and "open collector" TTL devices easy. See "Interfacing to Switches and other devices" below.

High current output at J3
Eight lines at J3 can be used as high current drivers. These outputs switch loads to ground. Outputs are controlled by Port B on the 82C55. Its address is 65.

Logic outputs from this port are inverted. That is, when a 1 is written to the high current port, the output is switched on and goes low.

The output driver chip, U20, can be replaced with a DIP shunt jumper so it is like the other lines at J3.

NOTE: Outputs at the high current lines are not compatible with TTL logic levels and should not be used to drive other logic devices.

Each of the high current outputs can sink 100 mA.at 50V.

Two lines from the high current port (Port B, 0 and 1) are used when the keypad is scanning 24 keys. These lines (at J3-8 and -10) may not be used for control purposes.

WARNING:
External supplies using the high current outputs must be tied to J3, pin 26 and NOT the power connector. Failure to do so can produce a ground loop and cause erratic operation.

The thermal time constant of U20 is very short, so the number of outputs that are on at any one time should include those that overlap even for a few milliseconds.

Incandescent lamps have a "cold" current of 11 times its operating current. Lamps requiring more than 50 mA. should not be used.

Protection diodes must be used with inductive loads. Refer to figure 6-2.

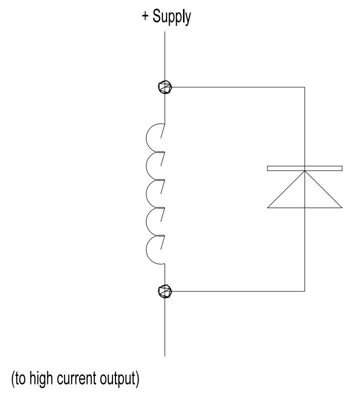


Figure 6-2 Inductive load protection

Do not parallel outputs for higher drive. This could result in damage since outputs do not share current equally.

High current output at P2
The P2 terminal marked “SWPWR” is a 2 Amp, 50V FET switch to ground. ON resistance is about 1 ohm. It is intended to switch back light power for the LCD graphics display. However, it can be used for any other purpose.

The switch is controlled by software as follows:

```
OUT &E7,1    :'Turns on switch
OUT &E7,0    :'Turns off switch
```

The switch is turned OFF when only on a hardware reset or watchdog timeout.

Use the circuit in Figure 6-2 when switching inductive loads.

Interfacing to an opto-module rack
J2 and J3 I/O lines interface to an MPS-8, 16, or 24 position opto module rack. Lines not going to an opto module connect to a screw terminal on the MPS-XX series boards. This feature allows you to connect switches or other TTL type devices to the digital I/O

lines. The MPS-XX series boards accept OPTO-22 G4 series or Grayhill G5 modules. See *Chapter 18, RESOURCES* , for a list of suppliers.

Use the OPTO command and to access and control opto modules. The LINE command is used to access individual lines on the STB-26 or MPS-XX rack.

A CMA-26 connects J2 and J3 on the RPC-2350 to the MPS-XX board. Cable length should be less than 2 feet for the 8 position rack and 18 inches for the 16 and 24 positions. Excessive cable lengths cause a high voltage drop and consequently unreliable operation. Be sure to supply + 5V and ground to the appropriately marked terminals.

You must configure the 8255 ports for outputs before using them. Use the following table to determine the corresponding opto channel for a particular 82C55 port:

Opto channels	**82C55 port**	**Connector**	**Addr.**
M0-M3	Lower C	J2	2
M4-M7	Upper C	J2	2
M8-M15	A	J2	0
M16-M23	B	J2	1
M100-M103	Lower C	J3	66
M104-M107	Upper C	J3	66
M108-M115	A	J3	64
M116-M123	B	J3	65

"Opto channel" is the position as marked on the MPS-xx board. The channel number is preceded by a 'M' character on the MPS board. When connecting J3 to an opto rack, add 100 to the number on the rack. J3 has a high current output on port A (channels M8-M15). Replace U20 with a shunt jumper to operate normally.

To turn on an opto module, an output line must be low. A module is turned off by writing a '1' to a channel. The logic at J3 port A, with the high current outputs installed is just the reverse. A '1' at a line causes the module to turn ON.

High current outputs at J3 port A are optionally configurable as TTL I/O by replacing U20 with a DIP shunt jumper. This keeps logic compatible with ports B and C. If opto channels 8-15 are used as inputs, then U20 must be replaced by a DIP shunt jumper.

Configuring digital I/O lines

Lines are configured during program execution using the CONFIG PIO command. On power up or reset, all lines are inputs.

When a line is configured as an output, it can sink a maximum of 2.5 mA. at 0.4V and can source a minimum of 2.5 mA.at 2.4V. When driving opto modules, the outputs sink 15 mA.at 1.0V.

Digital I/O programming example

The following example reads a switch at port A, bit 3 (J2-25), reads an opto module at channel 1 and writes an opto module at channel 5. A LED is controlled at J2-10 (port B, bit 0).

```
200 D = BIT(0,3)   :'Get status port A
210 F = OPTO(101)  :'Read opto module,
                      ch. 1
220 OPTO 103,ON    :'write module 3
230 BIT 1,0,0      :'turn on J2-10
240 BIT 1,0,1      :'turn off J2-10
250 A = LINE(103)  :'Reads pin 3 at J2
260 LINE 4,1       :'Set line # 4 to 1
```

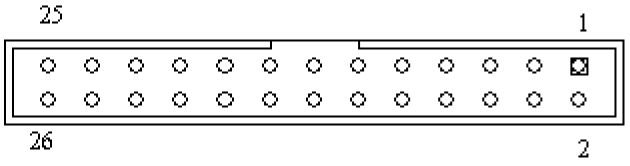


Figure 6-3 IDC pin out viewed from top

Connector pin out - J2

J2 Pin #	82C55 Port/bit	Description	Opto Channel
19	A/0		8
21	A/1		9
23	A/2		10
25	A/3		11
24	A/4		12
22	A/5		13
20	A/6		14
18	A/7		15
10	B/0		16
8	B/1		17
4	B/2		18
6	B/3		19
1	B/4		20
3	B/5		21
5	B/6		22
7	B/7		23
13	C/0	Lower C	0
16	C/1	Lower C	1
15	C/2	Lower C	2
17	C/3	Lower C	3
14	C/4	Upper C	4
11	C/5	Upper C	5
12	C/6	Upper C	6
9	C/7	Upper C	7
26		Ground	Ground
2		+ 5V	+ 5V

Connector pin out - J3

J3 Pin #	82C55 Port/Bit	Alternate function	Opto Channel
19	A/0	LCD port J6	8
21	A/1	LCD port J6	9
23	A/2	LCD port J6	10
25	A/3	LCD port J6	11
24	A/4	LCD port J6	12
22	A/5	LCD port J6	13
20	A/6	LCD port J6	14
18	A/7	LCD port J6	15
10	B/0	High curr. /Keypad	16
8	B/1	High curr. /keypad	17
4	B/2	High current	18
6	B/3	High current	19
1	B/4	High current	20
3	B/5	High current	21
5	B/6	High current	22
7	B/7	High current	23
13	C/0	Shared w/J5 keypad	0
16	C/1	Shared w/J5 keypad	1
15	C/2	Shared w/J5 keypad	2
17	C/3	Shared w/J5 keypad	3
14	C/4	Shared w/J5 keypad	4
11	C/5	Shared w/J5 keypad	5
12	C/6	Shared w/J5 keypad	6
9	C/7	Shared w/J5 keypad	7
26		Ground	Ground
2		+ 5V	

COMMANDS

The following table lists CAMBASIC commands used for digital I/O.

Command	Function
BIT	Function returns status of bit at an I/O address
BIT	Command sets a bit at an I/O address
CONFIG PIO	Configures J3 I/O port
INP	Returns a byte from an I/O address
LINE	Returns status of an opto line
OPTO	Sets an opto module output
OUT	Writes a byte to an I/O address
PULSE	Reads or writes a pulse at a port.

See also ON BIT, ON COUNT, ON INP, and related statements.

DIGITAL I/O WORKSHEET

Use the following tables to help you plan how digital lines will be used and referenced in your design. It will also help you spot potential conflicts with multiple use lines (such as keypad port on J3).

Copy these pages if necessary.

The following are the addresses for each of the 8 bit digital ports.

Connector J2. Addresses 0-2.
Port A = 0
Port B = 1
Port C = 2

Connector J3. Addresses &40-&42.
Port A = &40
Port B = &41
Port C = &42

J2 Pin #	82C55 Port/bit	Opto Channel	Description/use	Associated CAMBASIC variable, function, or task number
19	A/O	Example	*Start switch*	ON BIT 0,0,0 GOSUB ..START
19	A/0	8		
21	A/1	9		
23	A/2	10		
25	A/3	11		
24	A/4	12		
22	A/5	13		
20	A/6	14		
18	A/7	15		
10	B/0	16		
8	B/1	17		
4	B/2	18		
6	B/3	19		
1	B/4	20		
3	B/5	21		
5	B/6	22		
7	B/7	23		
13	C/0	0		
16	C/1	1		
15	C/2	2		
17	C/3	3		
14	C/4	4		
11	C/5	5		
12	C/6	6		
9	C/7	7		
26		Ground		
2		+ 5V		

J3 Pin #	82C55 Port/Bit	Other use for line	Opto Channel	What line is used for	Associated CAMBASIC variable, function, or task number
4	B/2	Example	Example	*Solenoid 1 on/off*	SOL1 = status, BIT &41,2,SOL1
19	A/0	LCD port J6	8		
21	A/1	LCD port J6	9		
23	A/2	LCD port J6	10		
25	A/3	LCD port J6	11		
24	A/4	LCD port J6	12		
22	A/5	LCD port J6	13		
20	A/6	LCD port J6	14		
18	A/7	LCD port J6	15		
10	B/0	High curr./Keypad	16		
8	B/1	High curr./keypad	17		
4	B/2	High current	18		
6	B/3	High current	19		
1	B/4	High current	20		
3	B/5	High current	21		
5	B/6	High current	22		
7	B/7	High current	23		
13	C/0	Shared w/J5 keypad	0		
16	C/1	Shared w/J5 keypad	1		
15	C/2	Shared w/J5 keypad	2		
17	C/3	Shared w/J5 keypad	3		
14	C/4	Shared w/J5 keypad	4		
11	C/5	Shared w/J5 keypad	5		
12	C/6	Shared w/J5 keypad	6		
9	C/7	Shared w/J5 keypad	7		
26		Ground	Ground		
2		+ 5V			

CHAPTER SYNOPSIS

- ● Initializing and using the RTC
- ● Y2K and Y2.4K issues
- ● Using clock interrupts

DESCRIPTION

The RPC-2350 has a battery backed Calendar/clock. When used in conjunction with the DATE$ and TIME$ commands, the current date and time can be set and read. It is accurate to 1 minute/month at 25°C and is not adjustable.

The clock data sheet is in the applications disk. See RTC72412.PDF.

SETTING DATE AND TIME

The clock must be turned on before it is used. This need be done only once. To turn on the clock, type:

```
CONFIG CLOCK 1
```

The date and time can be set while running a program or in the immediate mode. Date and time are treated as strings and not numbers. To set the date and time:

```
date$="04-19-99"
time$="13:56:00"
```

To retrieve date and time as part of a program:

```
2000 DA$ = DATE$(0)
2010 TI$ = TIME$(0)
```

You can also print the date and time in the immediate mode:

```
pr time$(0)
13:56:03
```

YEAR 2000 AND BEYOND

The clock on the RPC-2350G is year 2000 compliant under the following condition:

> Date is always returned as the last two digits of the year. The first two digits, “19" or “20" must be programmed into your system. As of the time this manual was written, this required a quick check on the year. If it was not “99", then you assumed it was some time in the 20's.

The clock will roll over on December 31, 1999 at 23:59:59 to January 1, 2000 with no problems.

CAMBASIC operating system does not use or need any real time clock values for its operation. A clock is not needed in order for CAMBASIC to operate.

The clock compensates for leap year in 2000. Should you expect this product to work into the 22nd century, it will add a leap day in the year 2100 also. This, of course, is not supposed to happen until the year 2400. If you think your program will be working in the year 2100, you will have to compensate for this by resetting the date when read as February 29, 2100 to March 1, 2100.

CLOCK INTERRUPTS

The RTC may be programmed to generate interrupts at 1 second, 1 minute, or 1 hour intervals. Longer interrupt intervals are convenient especially when ON TICK interrupts are running.

An interrupt is generated when the real time clock counters increment the unit of time selected for the interrupt interval. Most of the time, the first interrupt will be shorter than the interval period selected. For example, suppose you want to interrupt every minute. If the real time clock's seconds were at 45 , the first interrupt will occur in 15 seconds. Interrupts will then occur every minute. Operation is similar for hourly interrupts.

The clock is programmed to interrupt at specific intervals in software. See CLOCK1.BAS for example program.

Jumper W10 to enable interrupts.

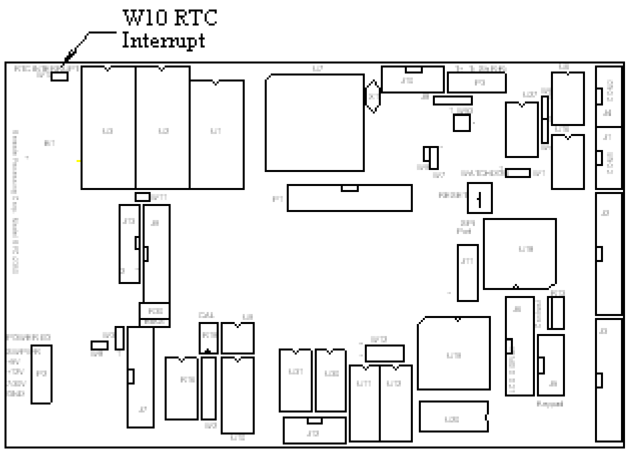


Figure 7-1 RTC interrupt jumper W10

Use the following table to set clock interrupt periods. "Value" is written to I/O port &14E.

Interrupt interval	Value
1 second	6
1 minute	10
1 hour	14

Write these values to address &14E to set the interrupt period.

```
OUT &14E,10  :'Set interrupt period to
               1 minute
```

Write a 0 to address &14D to clear any interrupts before executing ON ITR 0 and while in the interrupt subroutine.

```
OUT &14D,0   :'Clear interrupt
```

COMMANDS

The following is a list of CAMBASIC commands for the calendar/clock.

Command	Function
CONFIG CLOCK	Turns clock on or off
DATE$	Sets date
DATE$(0)	Returns date
TIME$	Sets time
TIME$(0)	Returns time
ON ITR 0	Responds to interrupt

CHAPTER SYNOPSIS

- Brief description of analog input capabilities
- Acquiring analog data
- High voltage interfacing
- Converting analog readings to real world units
- Calibration
- Analog output discussion
- 4-20 mA output
- Analog power supply

DESCRIPTION

The RPC-2350 has eight single ended or four differential analog input channels than can be interfaced to external analog devices. These channels can be used to measure voltages from transducers, 4-20 mA. current loops, thermistors, etc. The converter reads a voltage and returns a 12 bit (4096 count) number in under a milli-second. Inputs are programmable for 0 to + 5 or ±2.5 volt, single ended or differential mode.

Additionally, 2 analog output channels with 12 bit accuracy are optionally available. Output voltage is 0-5V, 0-10V, or ±5V. Outputs can drive optional 4-20 mA. current loops.

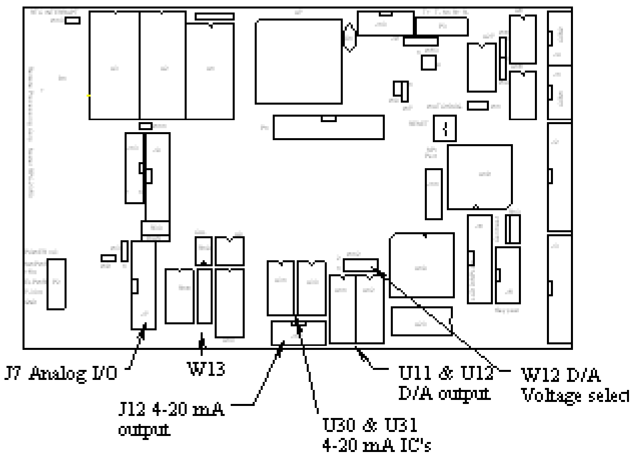


Figure 8-1 Analog connectors and jumpers

Filter capacitors may be added to pads designated as W13. This can reduce noise on analog inputs. Values are application dependent. 0.01 mfd is a good value to start from. Higher values may be used in extremely noisy environments or when time between samples is long (> 100 ms).

Input impedance is 100K ohm to ground. Inputs are protected to ±12V. Readings on other channels are affected when one channel is over range.

Conversion time is under 500 micro-seconds/channel.

AIN function is used to return a voltage while AOT writes an output voltage.

This chapter begins with basic hook-up information, then proceeds to initialization, data reading, and calibration. Analog output option is discussed near the end.

Analog output 1 (AOT 1) may optionally provide software contrast control for the LCD graphics display. When it is used for this purpose, 4-20 Ma output or other voltage output may not be used.

CONNECTING ANALOG I/O

Analog I/O interface via J7. The STB-20 terminal board may be used to bring signals to terminal blocks.

The following table defines the signal pin out from analog I/O port J7.

J7 Pin #	Signal
1	CH0 input
2-16	Ground (even pins)
3	CH1 input
5	CH2 input
7	CH3 input
9	CH4 input
11	CH5 input
13	CH6 input
15	CH7 input
17	DAC 0 output
18	+ 12V
19	DAC 1 output (also graphics contrast)
20	-12V

Initializing Inputs

The RPC-2350 can have up to eight single-ended inputs, four differential, or a mixture of single ended and differential inputs. On a reset, inputs are configured for 0-5V, single ended.

Initialization is performed using the CONFIG AIN command. The syntax is:

CONFIG AIN *channel, input, range*

Where:

channel is from 0 to 7 for single-ended or differential. Differential inputs require 2 lines and are specially paired as shown below. The channels you specify in a "mixed" application depends upon what lines are used for single ended and differential.

Differential inputs operate in a special way. Use the following two tables for differential inputs.

When ***channel*** = odd

Ch #	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Polarity	-	+	-	+	-	+	-	+
channel	1	1	3	3	5	5	7	7
J7 pin #	1	3	5	7	9	11	13	15

When ***channel*** = even

Ch #	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Polarity	+	-	+	-	+	-	+	-
channel	0	0	2	2	4	4	6	6
J7 pin #	1	3	5	7	9	11	13	15

For example, if you wanted one differential input, channel 0 would use J7 pin numbers 1 and 3. Single ended inputs 2-7 are available.

input specifies single ended or differential. 0 = differential, 1 = single ended.

range is voltage input. 0 = ±2.5V and 1 = 0 to + 5V.

Differential Mode

When differential mode is specified, inputs are actually pseudo-differential. What this means is that a ground reference is needed. For example, you cannot place a battery between channel 0 and 1 and get an accurate reading. The (-) input must be referenced to ground. An example of where pseudo-differential works is an output from a bridge network.

A pseudo-differential input subtracts the DC component from an input. The IC maker recommends the (-) input remain stable within 1 count with respect to ground for best results. Connecting a 0.1 uF capacitor from the (-) input to ground works well.

When operating in differential mode, relative + and - voltages must be connected to specific inputs. When inputs are reversed, a conversion returns a 0. When the relative voltage changes, perform a conversion on the alternate channel. CONFIG AIN must be performed on both channels to be valid.

Pairs of channels can be differential while others single ended. Thus, if channel 0 and 1 are differential inputs, channels 2-7 may be single ended.

Examples using CONFIG AIN

Below are sample syntaxes for the CONFIG AIN command:

1. Single ended mode, 0-5V input

```
CONFIG AIN chan,1,1
```

The input voltage is from 0 to 5 volts. The result from the AIN function is 0 for 0.000V and 4095 for + 4.9988V. **chan** may range from 0 to 7, if no other channels are used for differential inputs.

2. Differential mode, 0 to + 5V input

```
CONFIG AIN chan,0,1
```

chan can be 0, 2, 4, or 6. The input may range from 0 to + 5V. However, if the (-) input is more positive than the (+) input, the result will always be zero. The result from the AIN function is 0 for a difference of 0.000V and 4095 for a difference of 4.9988V.

3. Single ended, ±2.5V input

```
CONFIG AIN chan,1,0
```

The input range is -2.5V to + 2.5V. The result from an AIN function is 0 for -2.500V, 2048 for 0.000V, and 4095 for + 2.4998V.

4. Differential, ±2.5V input

```
CONFIG AIN chan,0,0
```

The input ranges from -2.5V to + 2.5V. The result is the difference of the two voltages. AIN will return 0 for a difference of -2.500V, 2048 for a difference of 0.000V, and 4095 for a difference of 2.498V.

ACQUIRING ANALOG DATA

Once the analog input is initialized, the AIN function is used to acquire data. The syntax is:

```
S = AIN(ch)
```

Where: **ch** = channel number, 0-7

This command reads the voltage and returns a number from 0 to 4095 to the variable S. The number returned corresponds to the voltage input and the type the channel was configured for.

To convert the returned numbers to a voltage, use the following formulas:

5V	Unipolar:	A = 0.00122 * AIN(channel)
±2.5	Bipolar:	A = 0.00244 * AIN(channel) - 5

The AIN function requires about 500 micro-Seconds to convert a channel of data. Additional time is needed to store the data. Saving data to a single dimension array takes 500 micro-seconds longer than saving to a simple variable.

Data logging on a timer tick

Some application require that data be taken at fixed intervals. The ON TICK construct can be used to take data in intervals from 0.01 to 655.35 seconds. The program below takes 100 samples on 2 channels every 10 seconds.

```
10 DIM F(100,2)
20 ON TICK 10 GOSUB 50
30 ..this is a dummy loop
40 GOTO 30
50 F(I,0) = AIN(0)
60 F(I,1) = AIN(1)
70 INC I
80 IF I = 100 THEN ON TICK 10 GOSUB
90 RETURN
```

Line 80 shuts off interrupts after 100 samples.

MEASURING HIGHER VOLTAGES

Input voltages higher than 5V are measured by placing a resistor in series with the input. Use the following formula to determine the required series resistance:

Rs = Vi * 20,000 - 100,000	0-5V range
Rs = Vi * 40,000 - 100,000	0-2.5V range

Rs is the resistor value in ohms in series with the input. Vi is the maximum input voltage. If the result of your calculation is 0 or negative, a series resistor is not necessary.

NOTE: When an input voltage exceeds its voltage range, other channels values are affected.

The 100K ohm resistor is R15. This is a 2% part. You may want to add a trim resistor in series with a fixed to obtain higher accuracy.

Since input impedance is higher, noise increases. A capacitor at the appropriate channel at W13 will reduce noise.

CONVERTING ANALOG MEASUREMENTS TO REAL WORLD UNITS

Inputs are converted to engineering units of measurement by performing scaling calculations in the program. The AIN function returns values from 0 to 4095. To change these numbers into something more meaningful, use the following formula:

var = K * AIN(*n*)

n is the analog channel to read. K is the scaling constant. K is obtained by dividing the highest number in the range of units by the maximum AIN count (4095). *var* result is in real world units (PSI, pounds, inches, volts, etc.)

Example 1: To measure the results of an A/D conversion in volts and the voltage range is 0 to 5V, divided 5 by 4095 to obtain K.

K = 5/4095
K = .001221

Your program could look something like:

```
1000 C = .001221 * AIN(N)
```

Example 2: You want to measure a 0 to 200 PSI pressure transducer with a 0 to + 5V output. Divide 200 by 4095 to obtain the constant K.

K = 200 / 4095
K = .0488

The result is in PSI when used as follows:

```
1000 B = .0488*AIN(0)
```

Noise Notes

An input channel can appear to be noisy (change readings at random) if unused inputs are allowed to float. To minimize noise (and increase accuracy), connect all unused inputs to ground.

A high impedance input is sensitive to voltage pickup. Noise is minimized by running wires away from AC power lines.

NOTE: Avoid running the cable over inductor L1. This can increase noise when using 7-30V input..

A low impedance voltage source helps to reduce noise pick up. Shielded cable can help reduce noise from high impedance sources. Make sure the shield is not used for power ground. Using the shield for power ground defeats its purpose. Try connecting the shield to ground at only one point, not at both ends. You may need to run a separate ground wire.

Wire pairs can also be twisted. 5-6 twists/foot provides a reasonable amount of noise cancellation.

Noise is defined in this section as any random change from a known input. The amount of noise you can expect under normal operating circumstances is ±3 counts for any input range. Under ideal conditions, noise contributes less than a count.

One way to compensate for noise is to take a number of samples and average the results. Taking 7 or more samples, in theory, cancels out any effects of noise. A problem with this is noise tends to group together. Taking 7 readings at one time might show no change from the norm. Another 7 readings might be all high. If possible, try to spread out readings over a period of time (several seconds if possible).

Jumper block W13 is used to install filter capacitors. Generally, the higher the source impedance, the lower the capacitor you will need. A 0.1 µF capacitor filters noise nicely when impedance is 100K. While installing capacitors filters noise, it also reduces the frequency response. How much depends upon your source impedance and capacitor values.

Noise is, by definition, random. If you were to plot out the deviations from a norm, it would roughly resemble a bell shaped curve. Experiments on the RPC-2350 have shown that over 99% of the readings are within the ±3 count reading. Noise readings were made with all inputs shorted to ground and with no cable connected to J7.

CALIBRATION

The A/D converter is calibrated using an external voltmeter. For 12 bit accuracy, you must use a voltmeter with an accuracy of 0.02% or better.

To calibrate the RPC-2350:

1. Connect the digital voltmeter ground to U10, pin 14. (Alternate ground pins are J7-2, 4, 6, 8, 10, 12 or 14).

2. Connect the digital voltmeter '+ ' lead to U9, pin 6.

3. Adjust trim pot R19 for 5.000VDC.

You may increase the reference voltage to a higher value, up to 5.12V. This will allow you to detect if an input device is at or above its 5V output range.

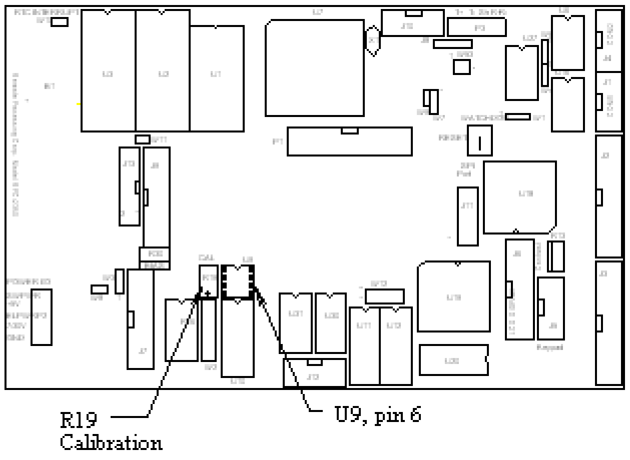


Figure 8-2 Calibration

ANALOG OUTPUT

Two optional analog output channels are independently configured for three voltage ranges. These ranges are jumpered in hardware. Refer to the following table for jumper settings. See Figure 8-4 for W12 detail.

Range (Volts)	J7-17 (AOT 0)	J7-19 (AOT 1)
0 to + 5	W12[2-4]	W12[1-3]
0 to + 10	W12[8-10]	W12[7-9]
-5 to + 5	W12[6-8]	W12[5-7]

Channel 0 output is J7, pin 17 and channel 1 is J7, pin 19.

Analog output 1 may optionally be used for software contrast adjustment for the LCD graphics display. When it is used for this purpose, analog output 1 may not be used for other purposes, including 4-20 Ma output.

IC Installation
The figure below shows where D/A IC's are installed.

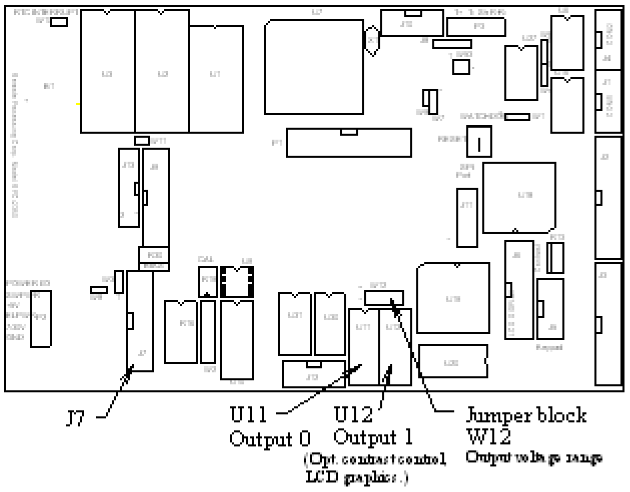


Figure 8-3 Analog output IC's, jumper, and connector

Analog output IC's are Analog Devices AD7248 type. This part may be ordered under Remote Processing Part number 1454.

Follow these steps to install analog output IC's.

1. Turn off power to the board.
2. Orient board as shown above. Orient IC so pin 1 (notch on IC) is towards the top of the board.
3. Install IC into appropriate socket. U11 for channel 0, U12 for channel 1.

4. Set jumper W12 for desired output voltage.

You are now ready to power up the board.

Programming voltage output
The AOT command is used to send data to an analog output. The syntax is:

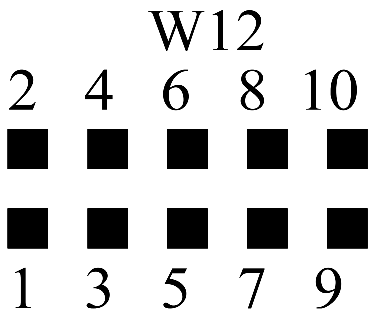


Figure 8-4 Jumper W12 detail

AOT ***channel, value***

Where:
channel specifies the analog channel to write data to

and can be either 0 or 1. ***channel*** 0 is on pin 17 and 1 is on pin 19.

value is the value to output from 0 to 4095.

Use the following table to convert from a desired voltage to a *value*.

Range	Formula
0 to 5V	*value*= V * 819
0 to 10V	*value*= V * 409.5
-5 to + 5V	*value*= V * 409.5 + 2047.5

The result of the formula produces a number which can be used in place of *value*.

Output Current
D/A output impedance is 0.5 ohms. Short circuit current is 40 mA. The analog power supply limits this current to something a little less. Practical maximum output current from a D/A is 10 mA.

Noise
Analog outputs generate noise in the 100KHz + frequency range. Many devices are not affected by this noise. However, if noise is a problem, put a capacitor (1 µF or so) on the output. Pads are provided at C23 and C24 near J7. C23 filters output 0 and C24 filters output 1.

4-20 mA. OUTPUT

Two 4-20 mA. outputs are optionally available. Interface is at J12. Current outputs are driven by the voltage DAC's. Configure each DAC used for 0-10V output to drive the current output.

Analog output 1 may be optionally used for software contrast control on the LCD graphics display. When it is used for this purpose, 4-20 mA output may not be used for this channel.

When you use a DAC to drive a current output, it cannot be used for voltage output.

Current output is proportional to the DAC voltage driving it. To program a current, you program a voltage using the AOT command. Thus, 0 V output supplies 4 mA. output while 10V output supplies 20 mA.

The following table lists J12 pin number, DAC, and current output.

Output No.	J12 pin	DAC driver	Current IC
0	2	U11 (AOT 0)	U30
1	10	U12 (AOT 1)	U31

The following table is J12 pin out.

Pin No.	Function
1	≈+ 12V, 40 mA. supply from RPC-2350
2	Current loop 0 output
3,5,6,9	Ground
4	Current loop 0 voltage power input
6	7-30V input from P2
8	Current loop 1 voltage power input
10	Current loop 1 output

Jumper W12 for 0-10V output for each channel you will use as current output. See table in Analog Output above for jumper instructions.

IC Installation
Current output IC's are installed in U30 and U31. See Figure 8-5 for location. The notch in the IC designates the top. Pin 1 is the upper left of the chip. Orient the board as shown in Figure 8-5 and install the chips in U30 for channel 0 and/or U31 for channel 1, keeping pin 1 to the upper left.

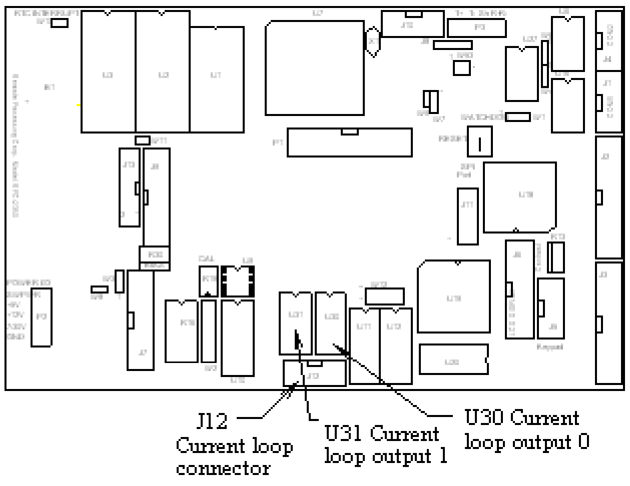

J12 Current loop connector
U31 Current loop output 1
U30 Current loop output 0

Figure 8-5 Current loop IC's & connector

Current loop power

The current output IC's require at least 12V DC to operate. The internal + 12V supply may be used. It is available at J12-1. Connect J12-1 to J12-4 and/or -8 to use the internal supply.

When runs are long using small gauge wire you may need to use an external + 18V to + 30V DC supply to power the current loop. If this supply is also used to power the board, you can optionally connect current loop input power pins CLP0 and/or CLP1 (J12-4 and J12-8) to J12-6. This pin also goes to P2-4.

ANALOG POWER SUPPLY

The RPC-2350 generates its own power for RS-232, LCD display, and analog I/O. Unregulated ± voltages are available at J7, pins 18 (+ 12V) and 20 (-12V).

+ 12V output can supply about 45 mA. of current total. Subtract 20 mA. for each 4-20 mA. output.

-12V output can supply about 15 mA. The D/A's use some current (1 mA.)

COMMANDS

The following is a list of CAMBASIC commands for analog input and output.

Command	Function
CONFIG AIN	Configures analog input for voltage range and mode.
AIN	Returns result of reading for a channel
AOT	Sends value to D/A converter

Use the table below to allocate input channels for your application.

J7 Pin #	Analog input channel	Description/use
x	n	*0-5V, pressure*
1	0	
3	1	
5	2	
7	3	
9	4	
11	5	
13	6	
15	7	

CHAPTER SYNOPSIS

● Operating information
● Multiple use note

DESCRIPTION

16 position keypads are plugged into keypad port J5. Keys are arranged in a 4 x 4 to 6 x 4 matrix format. A key is recognized when a row and a column connect. Up to 24 keys can be scanned.

CAMBASIC scans and debounces the keypad every debounce time. Debounce time is fixed at 40 ms. A key is debounced when it is down for two scans (80 ms). Keypad presses may be returned either as a number from 1 to 16 (1-24 in 24 position scan mode)or as an ASCII character. The ASCII character returned corresponds to those on Remote Processing's KP-1 keypad. Character assignments are changed using the SYS(8) function.

Keypads from Remote Processing simply plug into J5. Keypad cable length should be limited to 5 feet.

If the keypad port is not used, it may be used as a general purpose digital I/O port.

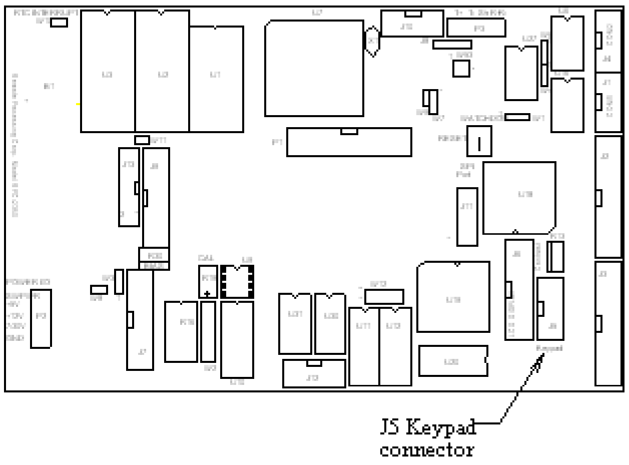


Figure 9-1 J5 keypad connector location

When 24 keys are scanned, U19 port B bits 0 and 1 are used for scanning. These lines also go to the high current buffer U20, and on to J3, pins 8 and 10. If you are using the high current port also, do not use these two lines.

PROGRAMMING THE KEYPAD

Sixteen and 24 position keypads use all of port C at U19. The 24 position keypad use and additional 2 lines from port B. Port B drives the high current sink, U20. If you are using the high current driver, or have replaced it with a DIP shunt jumper, lines at J3-8 and J3-10 are not usable with a 24 position keypad only.

U19 (keypad port IC) must be configured using the CONFIG PIO command. Some ports are optional, depending upon what you want to connect to it. Use the table below to help determine what a port should be (input or output) when using a keypad.

Port	Function	Configuration
A	LCD character driver General purpose TTL I/O	Output Output or input
B	High current sink When using 24 keypad Use 16 position keypad or general purpose TTL I/O	Output Output Output or input
LC	Keypad row (inputs)	Input
UC	Keypad column	Output

Check the table above to determine what you will be using. The Configuration column describes what that port should be set to.

The ON KEYPAD$ multitasking statement initializes the operating system to use the keypad. It tells the system what size of keypad to scan and what line to execute on a key press. When this command is executed, the scanning process begins.

INPUT KEYPAD$ allow you to input data from the keypad and echo the data to an LCD character or graphics display. Input can be a string or floating point number. Refer to INPUT KEYPAD$ command in the CAMBASIC manual. Use 8 for *echo port.* Only smaller characters can be echoed back to the display.

The KEYPAD$(n) function returns either the keypad character (as an ASCII value) or its position. When getting a character, keep in mind the difference between an ASCII value vs real. An ASCII '1' is not the same as the number 1 used for calculations.

The following example sets up CAMBASIC to scan a 16 position keypad. Ports A and B are set for outputs (presumably to drive the LCD display and high current port) The results are echo'ed to the display.

```
10 CONFIG PIO 1,0,0,1,0,64
20 'Optionally change keypad char 'B'
30 ' to the letter 'M'
40 POKE SYS(8)+7,77
60 ON KEYPAD$ 16 GOSUB 500
70 PRINT " Enter a number";
100 'loop for this example
110 GOTO 100

500 A$ = KEYPAD$(0)
510 IF A$ = "C" THEN ..clear_beep
520 IF A$ = CHR$(13) THEN ..enter
530 PRINT A$;
540 B$ = B$+A$
560 RETURN

600 ..clear_beep
610 B$=""
630 DELAY .4
650 PRINT CHR$(12); "    " CAR$(12);
660 RETURN

700 ..enter
710 FL = 1
730 RETURN
```

Program Explanation
Lines 10-80 set up the parameters for the keypad. Lines 500 to 730 process the key press. If a "C" or "#" is pressed, it is an exception and is handled that way. Otherwise, the character is displayed and stored.

Lines 700 to 730 process the "enter" key. The enter flag, FL, is set to a 1 to indicate to another part of the program that B$ has complete data.

The KEYPAD$(0) function returns a single character string that has been assigned to a particular key. Characters are assigned using the SYS(8) statement.

Keypad Commands
There are several keypad commands. See the table at the end of this chapter.

KEYPAD PORT PIN OUT - J5

The keypad port uses port C from an 82C55. Lower port C is configured as an input. Upper port C are outputs.

The table below lists J5's pin out, 82C55 port and bit, and its intended function.

Pin	82C55 Port/bit @ U19	Function
1	C/0	Row 1
2	C/6	Column 3
3	C/5	Column 2
4	C/1	Row 2
5	C/2	Row 3
6	C/4	Column 1
7	C/7	Column 4
8	C/3	Row 4
9	B/0	Column 5
10	B/1	Column 6

Ground is not used.

COMMANDS

The following is a list of CAMBASIC commands for the keypad.

Command	Function
CONFIG PIO	Configures digital I/O port
INPUT KEYPAD$	Input data from a keypad
KEYPAD$(n)	Returns last key from keypad port.
ON KEYPAD$ 16 ON KEYPAD$ 24	Causes a program branch when a key is pressed
SYS(8)	Returns keypad string address to modify characters.

CHAPTER 10 DISPLAY PORT

CHAPTER SYNOPSIS

- ● Differences between RPC-2350 and RPC-2350G
- ● Programming for a display
- ● Multiple use note

DESCRIPTION

A display, in conjunction with a keypad, can give an operator feedback on operation status and some level of control over the process.

There are two display ports on the RPC-2350G: J6 is for LCD character displays and J9 or J13 are for graphics displays. The RPC-2350 has only J6, used for LCD and VF character displays. This chapter discusses J6. See Chapter 15 for graphics display port.

The LCD character and graphic ports operate independently of each other. The LCD character port uses port A from an 82C55 PIO chip. These lines at J6 are shared with some on J3 also. The graphics port has its own driver and memory.

CAMBASIC commands are provided to position and write characters to each display. Additional commands are provided to draw lines, turn pixels on and off, and print large characters on the graphics display.

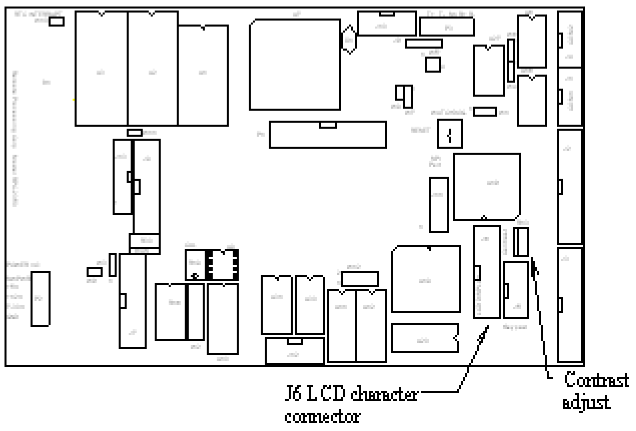


Figure 10-1 LCD character connector and contrast adjust

LCD CHARACTER PORT J6

You can use Liquid Crystal Displays (LCD) or vacuum fluorescent displays at J6. Display sizes range from 1 line by 8 characters to 4 lines by 40 characters.

The pin out at J6 is designed to plug directly into Remote Processing LCD 4 x 40 and LCD 4 x 20 displays. Simply plug these displays into J6. A contrast adjustment pot, R13, controls the viewing angle. This pot is adjusted after J6 is properly configured.

Any number of other LCD displays may be used. See the table at the end of this sub-section for cable pin out.

Configuring J6 for a display

Two lines of CAMBASIC code must be executed in the proper order before J6 is ready for displays.

First, you must configure the digital I/O port using the CONFIG PIO command. Port A must be configured as an output. If you are using a keypad, then set port C as shown in the example below. Port B is usually set to an output to drive the high current sinks. Refer to *Chapter 6, DIGITAL I/O*, for more information on port B and general programming information.

Put the following line of code in your program:

```
CONFIG PIO 1,0,x,x,x,64
```

'X' parameter is 0 or 1 as needed in your application. Refer to the *CAMBASIC Programming Manual* for more information about CONFIG PIO. The *address* for the display PIO chip is 64.

Next, determine the type of display you will be using. Refer to the *CAMBASIC Programming Manual* for a list of types under CONFIG DISPLAY.

The following example configures J6 for a LCD 4 x 20 display:

```
CONFIG DISPLAY 64,4,1
```

The cursor was selected as blinking.

There are two LCD character display demonstration programs that show how to position and write to the display:

LCD440.BAS	Writes to LCD 4 x 40
LCD420.BAS	Writes to LCD 4 x 20

USING TWO DISPLAYS

The RPC-2350G is not intended to use both character and graphics displays simultaneously. There is no provision for switching the software between two displays.

It is possible to write a LCD character driver in Basic. This routine will be slow and take up some space.

DISPLAY CONNECTOR PIN OUT

The display port uses an 82C55 for data and control. The table below lists a pin number and its intended function. A display may not use all lines even though they are available.

J6 Pin	82C55 Port/Line	Function WRT display (LCD displays)
1		+ 5V supply
2		Ground
3	A/4	~RS
4		Contrast Voltage
5	A/6	E1
6	A/5	R/~W
7		No connect
8		No connect
9		No connect
10	A/7	E2
11	A/1	DB5
12	A/0	DB4
13	A/3	DB7
14	A/2	DB6
15-20		No connect

The ~ character designates a logical NOT.

LCD character displays operate in 4 bit mode. Display lines DB0-DB3 are not connected.

J6 Pin	82C55 Port/Line	Function WRT display (VF displays)
1		+ 5V supply*
2		Ground*
3	A/4	D4
4		No connect
5	A/6	D6
6	A/5	D5
7		No connect
8		No connect
9		No connect
10	A/7	Strobe
11	A/1	D1
12	A/0	D0
13	A/3	D3
14	A/2	D2
15-20		No connect

VF character display connector table. Displays operate in 8 bit mode. Bring bit 7 on display to ground. Display bit 7 is not used.

***NOTE:** Due to high display current demand, it is recommended that separate + 5 and ground lines be brought to the display.

COMMANDS

The following is a list of commands used to control the displays.

Command	Function
CONFIG DISPLAY	Tells system type of display and initializes it.
DISPLAY	Core command to write to display for printing and positioning.
CONFIG PIO	Initializes digital port

CHAPTER SYNOPSIS

● Uses and limitations of sound/timer output
● Connecting to a speaker

DESCRIPTION

Sound may be used to drive a speaker or generate square wave pulses.

Sound timer and output line is used for other purposes. Do NOT use SOUND when using any of the following:

RS-485 communication
512K RAM memory(installed)
512K Flash memory

RS-485 uses the same timer as sound. CPU address line A18 is used to address RAM and Flash or provide the pulse output for SOUND.

SOUND Syntax is:
SOUND ***frequency***

frequency is from about 15 Hz to over 20,000 Hz.

Output is available only during run time. It is shut off in the immediate mode (i.e. Entering code.)

Frequency accuracy is dependent upon the CPU crystal of 18.432 MHZ. The factor that determines frequency accuracy and resolution is the basic timer frequency of 921.6 kHz. The timer is a 16 bit, meaning that there are 65,536 possible frequencies within the 921.6 KHz window. What this means is that while you might request a frequency, say 10,000 Hz, you will get something else. This is especially true at higher frequencies.

Sound output is available from J10-3. This output can go to a speaker or the counter at J10.

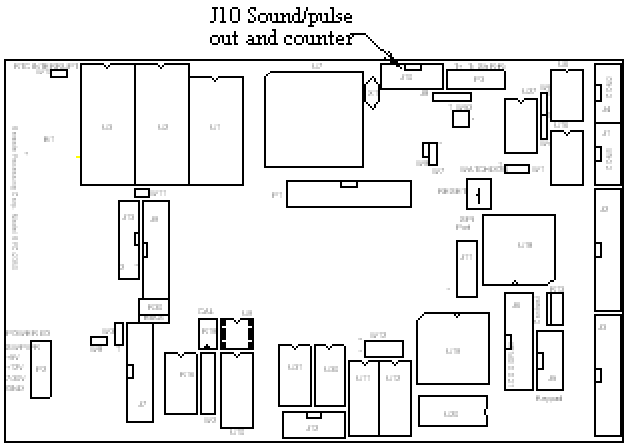


Figure 11-1 Sound/pulse output connector

CONNECTING TO A SPEAKER

Refer to figure 11-2 below for circuit connections to a speaker. The series resistor determines the volume. the Capacitor sets the lower frequency limit. Generally, values from 100 uF to 470 uF are adequate. The speaker may be any value but those with 50 ohms or greater produce higher db output.

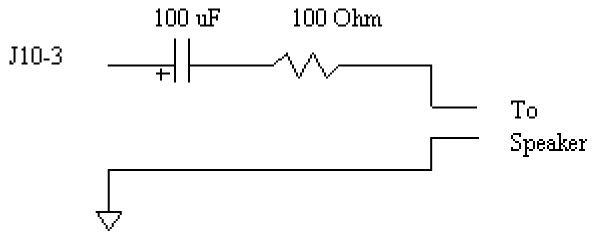


Figure 10-2 Speaker interface

CHAPTER SYNOPSIS

● Uses for a watchdog timer
● Cautions using watchdog

DESCRIPTION

A watchdog timer is used to reset the RPC-2350 if the program or CPU "crashes". When enabled, the program must write to I/O address &E8 to avoid a reset. The timeout is adjustable for 150 ms, 600 ms, or 1.2 seconds.

The watchdog should be disabled when using INPUT statements. Also, loops which do not end quickly or are of indeterminate duration should be avoided unless a timer reset pulse is included. An example of an indeterminate loop is one that waits for a port condition to change.

The watchdog is enabled by writing a 1 to address &E4, bit 0 and disabled by writing a 0 to the same location. The timer is reset by any access (read or write) to I/O address &E8.

The AIN command, in conjunction with the SPI port, access this address. Thus, executing either an AIN or SPI function also resets the watchdog timer, if enabled.

The watchdog timer is part of a voltage monitor and reset chip U14.

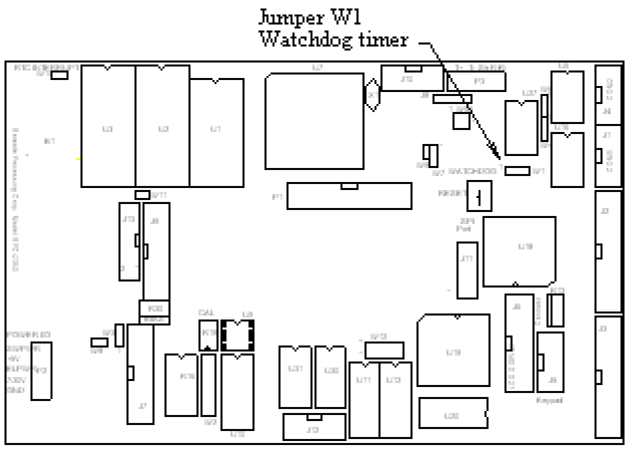


Figure 11-1 Watchdog timer jumper

Watchdog time is determined by jumper W1. Use the following table to set a timeout.

W1 Pins	Typical timeout	Range
[1-2]	1.2 sec.	500 mS to 2 Sec.
no jumper	600 ms	250 mS to 1000 mS
2-3	150 ms	62.5 to 250 mS

WARNING:

Once the watchdog timer is enabled, it can only be disabled by a BIT &E4,0,0 in the program. If the watchdog timer is running and the program stops for any reason (program error or < Esc> key hit), the card will reset. You can recover the program by typing UNNEW.

PROGRAM EXAMPLES

The following program fragments enable the watchdog timer, reset it while the program is running, and then disables it.

```
100 OUT &E4,1 :'Turn on watchdog
.
.
.
5000 OUT &E8,0 :'Reset timer
7000 A = AIN(N) :'Reset timer
.
.
.
10000 OUT &E4,0 :'Turn off watchdog
```

CHAPTER SYNOPSIS

● Discusses types of interrupts
● Interrupt priority

DESCRIPTION

Interrupts on the RPC-2350 can be broken down into two general groups: Hardware and software. Hardware interrupts are INT 0 and INT 1. Software interrupts include ON COM$, ON TICK, ON KEYPAD, and others that require software to execute.

Technically, timer and communications are also hardware interrupts. These are supported through software, and are considered software interrupts.

The NMI hardware interrupt is brought out to the expansion connector. It is not supported by CAMBASIC. It is accessible by assembly language.

INTERRUPT HANDLING BY CAMBASIC

Interrupt generating and handling is a bit complex. First, the general rule is explained then the exceptions to the rules are given.

When INT 0 and INT 1 lines go low, a CPU hardware interrupt is generated. Software responds by setting a flag. When the current CAMBASIC line is finished executing, the line number specified in ON INT is executed as a subroutine. Latency depends upon the complexity of the current CAMBASIC line being processed and when the interrupt occurred while the line was processed. Typical latency is about 1 mS.

Software interrupts such as ON BIT, ON KEYPAD, and ON INP require software processing. These routines scan I/O lines every system tick time (0.005 seconds). If a condition is met (keypad press, line changes status) a flag is set. Now, here is where things get a bit complicated.

The above interrupt tasks are checked every 8 program line statements (typically 0.005 seconds). This means that response to a line change could take an additional 5 milli-seconds from the time the event took place. Hardware interrupts are the exception. The operating system is forced to process these interrupts on the next statement.

What happens when interrupts occur simultaneously? There is an order of priority:

INT 0
INT 1
ON BIT 0 to
ON BIT 7
ON KEYPAD
ON TICK 0 to
ON TICK 2
ON COM$ 1
ON COM$ 0
ON COUNT 0 to
ON COUNT 7
ON INP 0 to
ON INP 7

If two interrupts happen simultaneously, they will be checked and started in this order. Unless you use the LOCK command, the next interrupt will not be processed for 8 commands. This means you can have an interrupt processing routine interrupt another, which can interrupt another. Since these are all subroutines, the number you can have active at one time is limited only by the amount of available RAM.

Interrupt service routines should be written as short as possible. Only those lines necessary to stop or start something should be processed. Heavy duty analysis and processing should be done in a non-time critical loop, if at all possible.

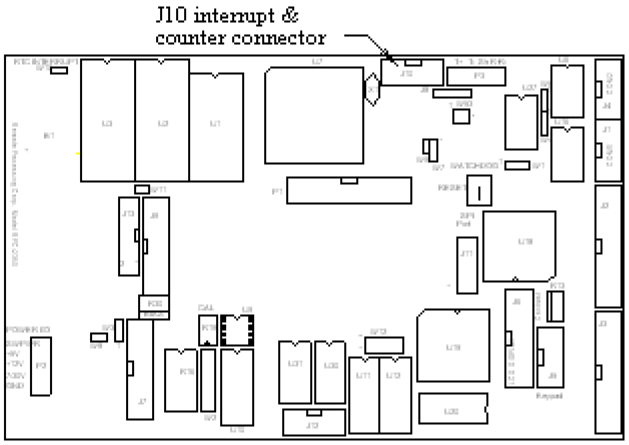


Figure 13-1 J10 and INT 1 location

HARDWARE INTERRUPTS

Hardware interrupts supported by CAMBASIC are INT 0 and INT 1. The real time clock uses INT 0, jumpered via W10. (See *Chapter 7*). The counter/quadrature encoder uses INT 1 (See Chapter 14). INT 1 also goes to J10-2 for external interrupts. Be sure to remove any jumpers from W7 if using external interrupts. INT1 is tied to a 10K ohm pull up resistor.

The ON ITR n GOSUB *line/label* tasking statement is used to initialize interrupts 0 and 1.

INT 0 and INT 1 are level sensitive. As long as the line is low, an interrupt is generated. This can be a blessing and curse, depending upon the application. Make sure you turn off or reset your external hardware interrupt source before executing a RETURN ITR n.

Hardware service routines require a RETURN ITR n at the end of the subroutine. This will re-enable the particular interrupt. If, for some reason you do not want the interrupt again, you can just RETURN. Some applications may actually require a delayed RETURN ITR. This is true if you are monitoring a slow moving pulse at the counter. When an interrupt is generated, the low signal output may not go away by the time the interrupt is finished. You can simply set a flag to let the main routine know later to re-enable the interrupt when the condition is gone. See CNTR2.BAS for an example.

Later you can execute RETURN ITR n to re-enable it.

The CLOCK1.BAS routine shows how interrupt 0 is used.

INT1 is available at J10-2. It is active low, level sensitive, and is used in conjunction with the counter carry or borrow output or an external signal applied to J10.

NOTE: Interrupts are frequently turned off while CAMBASIC runs certain time critical code. Times when interrupts are turned off include graphic display writing (when sparkle is on), brief periods when servicing communication interrupts, and writing to Flash. Normally a pulse 10 micro-Seconds wide can cause an interrupt. However, if you are saving to flash or to the graphics screen, the pulse should be 100 micro-Seconds to ensure capture.

NOTE: Interrupt 0 and 1 are level sensitive. As long as the line is low, another interrupt will be called as soon as RETURN ITR is executed. Make sure you clear the source of the interrupt before executing RETURN ITR.

SOFTWARE INTERRUPTS

Software interrupts are all other "ON" types. These interrupts look for an interrupt condition in software. The ON BIT, ON INP, ON KEYPAD$, ON COUNT, and ON TICK routines either scan or count first then determine if an interrupt should be declared. All of the above routines operate on a 5 mS interval. That is, every 5 mS lines are scanned, counters checked, and so on.

Commands that look at digital I/O lines, such as ON BIT, require a stable input condition for at least 5 ms in order to be recognized.

All of the subroutines use a simple RETURN to continue execution from where it was interrupted from.

See the examples in the CAMBASIC manual for more information on using these tasking statements.

COMMANDS

The following commands are used for interrupts in CAMBASIC.

Command	Function
ON BIT	Interrupt on line change
ON COM$	Interrupt on serial data
ON COUNT	Interrupt when count reached
ON INP	Interrupt on bit mask
ON ITR	Hardware interrupt
ON KEYPAD$	Interrupt on key press
ON TICK	Periodic interrupts
RETURN ITR	Return from hardware interrupt.

CHAPTER SYNOPSIS

- ● Brief description of the counter
- ● High voltage input and level sensing adjustment
- ● Use in program
- ● Measuring pulse width
- ● Measure frequency

DESCRIPTION

The RPC-2350 has a programmable high speed counter or quadrature encoder. The 24-bit counters are capable of up/down, binary, divide-by-n, and quadrature inputs. Count frequency is DC to 20 MHZ. The type of counter is an LSI Computer Systems LS7166. Its data sheet is in *Appendix A* and on disk as LS7166.PDF

COUNT(8) is used to read the counter. The OUT command is used to write and program the chip.

An interrupt, using ON ITR 1, may be detected on a carry or borrow.

A high voltage input, such as a signal from a proximity sensor, may be connected to one of the inputs.

Signals connect via J10. All input lines are pulled high through 10K input resistors. A quadrature encoder may be connected directly to J10.

A count is incremented when the signal at the 'A' input goes from low to high.

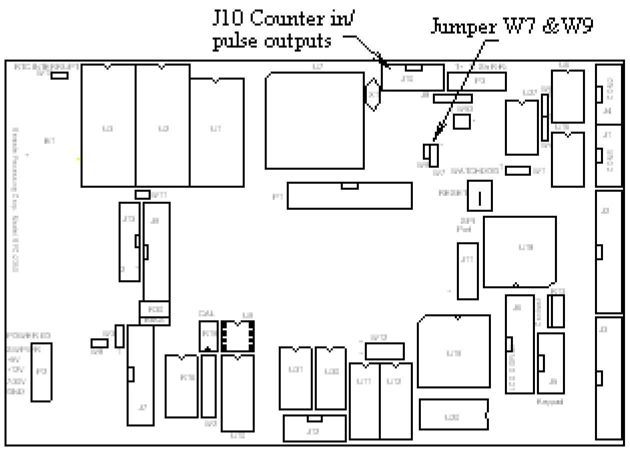


Figure 14-1 Counter and jumper location

COUNTER INPUTS AND OUTPUTS

The counter chip has four inputs and two outputs. Reference is made to the LS7166 counter registers. These registers are in *Appendix A* at the end of this manual.

Two of the inputs, designated as A and B, are counter inputs. The ICR (input control register) controls the function of these inputs. Encoders, switches, and other such devices are connected to these inputs. These inputs are very high speed. If you are going to use a mechanical switch, it is best to debounce it first or use the high voltage input described below.

The 'A' input (J10-9 or J10-10 through buffer) operates as up and down count input and a quadrature input.

The 'B' input (J10-8) can act as a down count input, direction control for input A, or a quadrature input.

Another input is LCTR (J10-6). It can load the counter or output latch. The ICR register controls the function of this line. If using it to control the output latch, you must read each register individually and not transfer the counter to the output latch as is done by COUNT(8). See CNTR5.BAS.

The ABGT input (J10-4) enables/disables A/B inputs or resets the counter. The ICR regsiter controls the function of this line. Normally, it does not have to be accessed.

The two outputs, CY and BW are counter carry and borrow signals. They are use to generate an interrupt (INT1) when the counter goes either through 0 or a preset. These outputs are controlled by the OCCR register. Status is read at the OSR register.

Interrupt selection
Jumper W7 can be used to interrupt the CPU on a counter carry, borrow, or external interrupt. Jumper W7[1-2] to interrupt on a carry (counter overflows). Set jumper W7[2-3] to interrupt on a borrow (counter underflow). Leave W7 open if using an external interrupt. INT1 goes to J10-2 for external interrupts.

See Figure 14-2 for W7 jumper pin out.

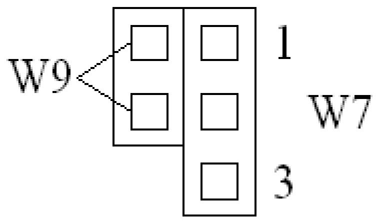


Figure 14-2 W7 and W9 jumper detail

HIGH VOLTAGE INPUT

Connector J10, pin 10 can accept a ±15V signal. As shipped from the factory, it detects a high input level (output goes low) at about + 3 volts and a low input (output goes high) at about + 2 volts. This level is programmable by changing R28.

Lower the value of R28 to increase the "high" level. For example, changing R28 to 23K raises the high input level threshold to about 4.2 volts and the low level to about 3.2 volts. The thresholds are approximate and change from lot to lot.

The output of the buffer connects to a counter input via jumper W9. When W9 is jumpered it will goto the counter's "A" input and to J10-9.

The buffer inverts the input signal. A count increments when a signal goes "high to "low" on this input.

This line is useful for connecting proximity switches to the counter. It may be used to filter switch contact closures by tying a capacitor from its input to ground. A 10K pull-up resistor is connected to the input.

PROGRAMMING

The LS7166 is capable of several operating modes, all of which cannot be discussed. See *Appendix A* for this chips operating modes. What are shown are example of how to program this chip and some common operating modes..

The counter chip must be initialized before using the COUNT function. You need to write to the ICR (Input control register), OCCR (Output control register), and possibly QR (Quadrature register) in order to set up the counter. Examples are given below for different operating modes.

The COUNT function returns the current counter value. Specifically, CAMBASIC writes a 2 to the MCR (Master Control Register), reads the 3 counter bytes from the OL (Output latch), and converts it to the proper internal BASIC format.

The LS7166 has several count related registers. The PR (Preset register) is a kind of holding register. A bit in the MCR (Master control register) transfers the information to the CNTR (counter). The PR is used to pre-load the counter. This pre-load value can be put into the main counter by setting a bit in the MCR or bringing the LCTR (J10-6) line low momentarily.

The CNTR is read by first setting another bit in the MCR to transfer CNTR to OL (Output latch). CAMBASIC COUNT(8) function does this.

The counter is capable of generating an interrupt every time CNTR equal PR, or when CNTR passes through 0 while counting up or down.

NOTE: Pulses from the LS7166 CY and BW pins must be long enough for the RPC2350 to recognize an interrupt. Counting speed is limited to about 100 KHz when interrupts are desired. Even these can be missed as CAMBASIC shuts off interrupts at points in the program, especially when writing to the graphics screen or Flash.

WARNING:

Do not use the CY or BW pulses to generate an interrupt in quadrature mode. The pulses are far too short and are easily missed by hardware. Contact Remote Processing for solutions.

Program Examples

This code, in CNTR1.BAS, resets the counter and enables the inputs. If desired, connect J2-19 to J10-9 to see the count increment. The count is printed once a second. If desired, you can bring J10-9 to another device.

```
10 pr "Counter test / demo program"
20 pr "Uses J2-19 to generate pulses."
30 pr "Connect to J10-9 (counter
input)"
40 print "Count continues until there
is an error. (about 16 million)
50 pr "Current count is printed every
second."
100 config pio 0,0,1,1,1,0    :'make
port A output
110 out &f1,32 :'reset counter to 0
120 out &f1,72 :'Enable counter A/B
```

```
inputs to count
130 on tick 0,1 gosub 1000
140 c = 1   :'initialize loop counter
200 bit 0,0,0
210 bit 0,0,1  :'rising edge increments
count
220 a=count(8)
230 if a <> c then print "Count error"
: end
240 inc c
250 goto 200
1000 print count(8)      :'show current
count
1010 return
```

The following example returns a frequency. Input signal is at J2-9 (A input)

```
10 OUT &F1,32  :'RESET COUNTER
20 OUT &F1,72  :'ENABLE INPUTS
30 ON TICK 0,1 GOSUB 1000

40 GOTO 40  :'HANG OUT HERE

1000 A = COUNT(8)     :'GET COUNT
1010 C = A-B   :'FIGURE CHANGE FROM
LAST TIME
1020 PRINT "Frequency = ";a
1030 B = A
1040 RETURN
```

The first frequency read will be off, due to initialization. Accuracy is increased by stretching reading to every 10 seconds. Other factors affecting accuracy include serial communications and other interrupt processing.

The counter will not miss counts. Due to interrupt latency, some counts will be larger than others. It is several counts off at about 8 kHz. If you average the counts it will be accurate.

Another problem with this routine is periodically, a large negative number is returned. This is because the counter has rolled over. This is corrected by periodically resetting the CNTR.

Program CNTR2.BAS sets up the LS7166 to cause an interrupt when a preset number of counts is reached. W7[2-3] is jumpered to interrupt on a borrow. To reload the count, bring the LDCTR line (J10-6) low. When the count is 0 again, another interrupt is generated. You can also count up provided you bring counter line 'B' (J10-8) low while counter line 'A' is high.

CNTR3.BAS interfaces to a quadrature encoder in x1 mode. The counter is pre-loaded to 100.

NOTE: See CAMBASIC resolution limit below.

CAMBASIC resolution limit
CAMBASIC stores numbers to 7 digits + exponent. The counter outputs numbers to 8 digits. This means that when the counter counts down from 0 to 1677215, CAMBASIC will store it as 1.67721E+ 7. The last digit is dropped.

You can compensate for this easily by introducing an offset. Preload the counter to some number, say 100,000. This becomes the zero point. When the count is below this number, the counter is in "negative" territory. See CNTR4.BAS

Program CNTR5.BAS reads the counter in Basic (not using COUNT(8) and prints the value in hex format. This routine can be useful when an external device triggers the LCTR line (J10-6) to transfer the count to a latch. The count at that time can be read.

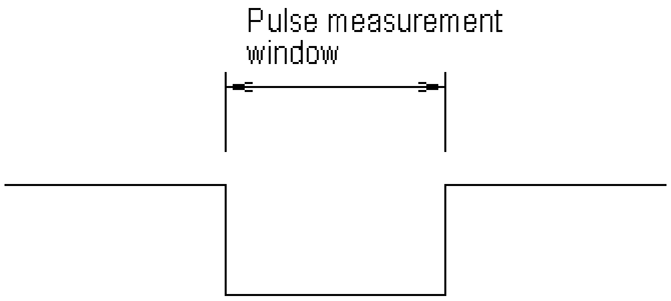
MEASURING PULSE WIDTH

You can measure pulse widths with 217 nano-Second precision. Widths can be as long as 3.64 Seconds using the counter input at J10.

There are limitations to measuring pulse widths. Below lists the major ones.

The pulse repetition rate must be slower than the time it takes CAMBASIC to respond to it. As a guide, the pulse repetition rate should be less than 100 Hz. Measuring a 50 micro-second signal every second is easy. Measuring a 500 micro-second signal every milli-second is difficult, if not impossible.

Only logic low pulses are measured. If a high pulse width is desired, invert the input signal. See figure below.



Signal levels are all TTL logic (0 to 5V).
The following signals at J10 are used to measure pulse widths:

J10 pin	Description
4	Counter gate. Measures when low.
7	4.608 MHZ clock output. Tie to J10-9
9	Clock input. Tie to J10-7

The LS7166 ICR register is programmed so input A (J10-9) is up count input and GATE input (J10-9) acts to enable inputs A/B when low.

If desired, LOAD input (J10-6) can be used to reset the counter. If this is desired, make sure the OL register is programmed for 0.

See the demonstration program CNTR6.BAS for a working example.

Basic operation is a follows:

Set the counter to 0
Read the counter and wait until it stops changing after it starts
Read the counter
Multiply result by 2.170139E-7

The result is the actual time.

J10 Pin out
The following is the pin out for J10.

J10 pin	Function
1	Ground
2	4.608 MHZ clock output
3	Sound output
4	Gate input
5	+ 5V
6	Load counter
7	INT 1 input
8	B counter input
9	A counter input
10	High voltage input

COMMANDS

The following commands are used with the multi-mode counter.

Command	Function
COUNT(8)	Reads multi-mode counter
ON ITR 1	Interrupt tasking
INP	I/O port read
OUT	Write to I/O port

CHAPTER SYNOPSIS

- General display information
- Connect a display
- Display modes
- Printing text
- Make and load fonts
- Making, saving and loading screens
- Drawing points and lines
- ancillary screen control
- Load and save graphic screens
- Touch screen positioning
- Cable and wiring diagrams
- Command summary
- Application programs

DISPLAY INFORMATION

The RPC-2350 initializes the display controller for the following type on power up:

Optrex DMF50174 320 x 240 pixel LCD
Planar EL320.240.36 320 x 240 pixel EL

The LCD plugs into J9. The EL display plugs into J13.

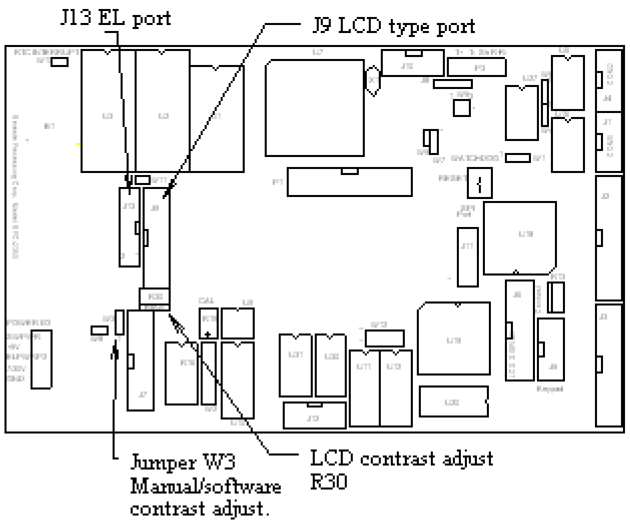


Figure 15-1 LCD and EL connector location

There are 3 character sizes available. Default character set is 5 x 7 pixels. This set displays 30 rows of 40 characters / line. Most American ASCII characters are displayed. The '\' character is the yen symbol. ASCII values 126 and 127 are displayed as → and ← respectively.

Medium size is 10 x 16 pixels (0.175" height). All printable ASCII characters are available. This set displays a maximum of 15 rows x 32 characters wide.

The largest size is 32 x 48 pixels (0.5" height). All ASCII characters are available. This set displays a maximum of 6 rows x 10 characters wide.

Medium and large characters are treated as graphics. They can be displayed in normal or reverse (light border, black character) mode. Fonts are stored in Flash EPROM. You may change fonts as desired. See "Changing and loading fonts" later.

Small character cursor position starts in the screen's upper left corner. Row, Column coordinates start at 0,0 and end at 45,29.

The controller is programmed for two planes. The first plane is small characters only. The second plane is larger characters and graphics. Each plane can be independently turned on or off or flashed.

Display snow or sparkle
Snow, or sparkle, were defined in the PC world as extraneous flashes of light on the screen. Sparkle appeared during updates on PC's. A similar phenomenon can manifest itself on the RPC-2350G. Instead of flashes of light, there are black or grey bars randomly running around an LCD screen. These are most noticeable in reverse character display mode. Flashes of light are even more noticeable on EL displays.

Sparkle is reduced by writing to controller memory during the "update" time. Unfortunately this tends to slow display updates considerably. Sparkle affects the RPC-2350G only when drawing medium and large size characters. It is especially noticeable when printing in reverse.

Update time can be critical. Typical times for printing 5 large characters is about 0.1 second with sparkle suppression. Without sparkle, the same characters are printed in about 0.020 seconds. Medium size characters take 2 times longer to print with sparkle suppression than without.

Default CAMBASIC mode is minimum sparkle. Sparkle suppression can be temporarily turned off. This is done by writing a 1 to the sparkle flag (using SYS(14)). Use the following program examples.

```
POKE SYS(15),1,0  :'Fast- w/ sparkle
POKE SYS(15),0,0  :'Slow- no sparkle
```

There is an unfortunate paradox because of this. Sparkle is most noticeable when displays are updated frequently (5 times/second). The problem is when you want them updated without sparkle, it takes more time.

NOTE: Sparkle suppression (default CAMBASIC mode) can be a problem when writing lots of text and when performing multitasking. Writing text for 1/10 second means during this time, all other interrupts are put on hold for that time (interrupt latency). You may have to sacrifice screen clarity for speed.

Sparkle may not be objectionable in your application. It is most noticeable on EL displays and when updating a LCD screen with reverse (white block with black lettering) characters.

Sparkle suppression is not completely eliminated when enabled. You will still see random bars flickering around the screen.

CONNECTING A DISPLAY

Operating a display is as simple as plugging the display cable into the appropriate connector. See Figure 15-1 above for connector locations. The RPC-2350G automatically initializes the controller on power up. A blinking cursor is displayed in the upper left corner of the screen.

Make sure the board works as described in Chapter 2, Setup and Operation, before connecting any display.

LCD Display
The LCD display plugs into J9. Back light inverter is connected separately to the + 5V and GND terminals on P2.

The Back light inverter may be powered ON or OFF under software control by connecting its ground to P2 terminal marked "SWPWR" (P2-1). This is a high current switch to ground. By default, it is off. To turn on, execute the BASIC statement:

```
OUT &E7,1
```

To turn the inverter off, execute

```
OUT &E7,0
```

Initially connect the Back light inverter ground lead to "GND" on P2 or your power supply.

Orient the display so the back light cable (2 wire) and lamp are on the right side, when viewing the display from the front.

When you power up the board, the graphics controller is initialized to display a blinking cursor in the upper left corner. You should see a white light coming from the back light lamp on the right side.

The back light requires some warm up time (about a minute) in order for the display to be readable. Adjust contrast pot R30 for optimal viewing contrast.

EL Display
The EL display connects to J13. Connection is one to one using a 2 mm ribbon cable.

External + 12V power must be applied to P2 terminal marked "ELPWR". This is necessary for display power. You can connect this same + 12V power to the "7-30V" terminal to power the board, if desired.

When you power up the board, the graphics controller is initialized to display a blinking cursor in the upper left corner

Verify operation - both display types
Chances are if you have the blinking cursor, the display is going to work. A quick way to verify operation is to type the following line in the immediate mode.

```
DISPLAY "Hello world"
```

The message should be displayed on the top line. The cursor should be blinking on the next line down.

You can execute any of the display commands in the immediate mode as well while running. For example, if you want to draw a lighted box, execute:

```
DISPLAY F(100,100),(120,120)
```

A list of graphics programs is shown at the end of this chapter. Download them to see how they display and are programmed.

DISPLAY LAYERS

There are two display layers: graphics and character. Each layer is 'OR'ed' with the other, meaning that a lighted pixel (or block) on one layer can obliterate another.

This CAMBASIC can display 3 character sizes. The 2 larger sizes are considered a graphic.

Each layer can be turned on, off, or flashed.

CONTRAST ADJUSTMENT

There are two contrast adjustment methods for LC displays. Both use the BIAS pot R30.

Jumper W3 determines if contrast is set only by BIAS pot R30 or can be also modified in software. Software control is handy if the display is subjected to wide temperature variations.

Software control uses analog output channel 1. If this channel is used for contrast adjustment, then analog output voltage and 4-20 mA. current are not available for this channel.

Mechanical Contrast Adjustment
Contrast set by R30 is factory default. W3[2-3] sets this condition. See figure 15-2 below for jumper detail. Adjust R30 BIAS pot for optimal viewing.

Software Contrast Control
Contrast can be controlled by software using analog output channel 1.

The contrast should adjusted manually at first. Simply power up the display and run one of the display programs. Adjust R30 contrast for optimal display. Then turn off the power and set the jumpers shown below. (Contrast voltage at display connector J9-5 is about -18 volts.)

Set the following jumpers:

W3[1-2]
W12[5-7]

Jumper W3 is set for software control. See Figure 15-2 below for jumper detail. Jumper W12 is set for +/-5V output from the D/A.

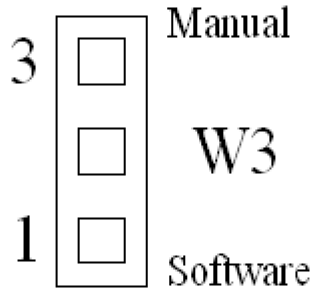


Figure 15-2 Jumper W3 detail

After the board is powered up, execute the following CAMBASIC statement:

```
AOT 1,2048
```

This sets the D/A output to 0V. This code should also be placed in the initialization section of your program.

The D/A has 12 bit resolution. This translates to 4096 possible voltages from -5 to +5V, or about 2.4 mV/step. This resolution is far too fine to be noticed. Noticeable changes in contrast start in steps of 200 counts. Using the AOT command, you can step up or down to increase or decrease screen brightness.

The following examples show a relative screen change for different voltage values.

`AOT 1,3000`	Decrease brightness
`AOT 1,2000`	Increase brightness

At some point an increase in brightness swamps out the contrast.

PRINTING TEXT

The GRAPDEMO.BAS program shows the different ways of printing all text characters and graphics. Run this program to see how it works.

NOTE: Medium and large characters use Flash EPROM in U3 to store fonts. Make sure W11 is installed.

There are 3 text sizes: Small (2.88 mm or 0.11" tall), medium (5.76 mm or 0.22" tall) and large (17.28 mm or 0.68" tall) (sizes are based on 0.36 mm dot pitch). Standard is small. This font is built into the graphics controller.

All fonts are fixed space, not proportional. Smallest font is 8 x 8 pixels. Medium font is 10 wide by 16 tall. The largest font is 32 pixels wide by 48 tall.

Small characters are printed on its own plane. Thus, small text can be turned off if desired. The larger sizes are considered graphics and are printed on the graphics plane.

Printing small characters is much like printing to a terminal. Unless there is a semi-colon (;) at the end of a PRINT or DISPLAY statement, the cursor advances to the beginning of the next line simulating a < CR> < LF> sequence.

Text is printed using a number of CAMBASIC commands. Some of them are listed below.

```
PRINT #10,"Text"
PRINT #10,USING "##.####";A
DISPLAY "Text"
DISPLAY (Row, Column); "Text"
DISPLAY L (Row, Column);"Text";
DISPLAY M (Row, Column);"Text";
DISPLAY M, R(Row, Column);"Text";
```

PRINT #10 can write formatted numbers and text in the same way as PRINT. PRINT uses small text. Larger characters must use DISPLAY. PRINT USING is not available for larger characters.

Medium and large characters are formed in CAMBASIC and are treated as graphics. Printing these characters is much the same as small ones. The major difference is CAMBASIC does not re-position text on the next line if you run out of room on the current line.

Positioning text
Small character text is positioned using the DISPLAY command. Text begins in the upper left corner at (0,0). The lower right corner is (29,39). Text is auto-incremented to the next position. When small text is at the end of the line, it is positioned at the next line. An entire screen of small text will scroll up one line when the bottom line is printed followed by a < LF> character.

Medium sized characters are positioned based on graphical X and Y pixel position. When printing a string, characters automatically advance to the right by 10 pixels.

The X,Y coordinates in the DISPLAY command for medium characters specify the upper right corner of the character block. Thus, a DISPLAY M(2,3) command starts printing the character on the 3rd pixel to the right, and 4th pixel down from the top (coordinates start at 0,0).

The largest character is positioned based on pixel and small character resolution. The X position starts on the column based on the small character set. Its range is 0-34. The Y position sets the top of the character. Its range is 0 to 192. Thus, you have 192 vertical points and 35 horizontal points to position a large character.

Medium and large characters are drawn as graphics. This means they appear on the graphics layer. The fonts for these characters are stored in Flash EPROM, U3. If this EPROM is missing or W11 is removed, the larger characters will display garbage. Fonts can be modified as desired. See "Changing and loading fonts" later in this chapter.

Printing normal and reverse characters
Medium and large characters may be printed in normal (white on black background) and reverse (black on white background). This is done by specifying the "R" parameter in the DISPLAY command.

```
DISPLAY M,R(0,0)"REVERSE";
DISPLAY M(0,16)"NORMAL";
```

The semi-colon (;)
You may (or may not) notice a semi-colon (;) after some, but not all DISPLAY and PRINT #10 statements. A (;) suppresses a < CR> < LF> sequence to the display (and serial ports). A comma (,) works in a similar fashion except a number of spaces are printed while it tabs to the next location..

The display driver was designed so when printing small characters, the screen acted like a terminal display. Thus, a < CR> < LF> sequence simply moved the cursor down one line.

Medium and large characters do not operate this way. A < CR> or < LF> are simply treated as spaces. Unless you include a ";" at the end of a DISPLAY M or DISPLAY L type line, you will effectively print two more blocks of characters. This is especially noticeable in the reverse display mode.

Scrolling
Small text automatically scrolls up 1 line when it reaches the bottom of the screen. Thus, the display acts like a terminal.

Scroll up time may be an important factor. It takes about 65 mS to move all 1160 characters of the text up one line.

CHANGING AND LOADING FONTS

Medium and large size character fonts are stored in Flash EPROM U3. You can change or customize the fonts using the supplied programs and a Windows bitmap graphics program (such as PC Paint Brush).

If U3 is missing, damaged or W11 is removed, garbage characters will be displayed.

You can change and load fonts using the programs supplied in the demo disk. They are under the FONT directory. Supplied are all of the .BMP files used to create each character. They are named as Lxxx.BMP or Mxxx.BMP. An 'L' prefix specifies a large character and an 'M' prefix is for medium. The number following this letter is the ASCII value. Thus, M65.BMP is a medium sized letter 'A'

Modification instructions are in a README file in the directory. Read this if you want to modify the characters..

Characters sizes are fixed at 10 x 16 for medium and 32 x 48 for large.

DRAWING POINTS AND LINES

There are two basic commands used for drawing lines and points:

```
DISPLAY LINE
DISPLAY P
```

Both commands have a counter part to clear lines and points:

```
CLEAR DISPLAY LINE
CLEAR DISPLAY P
```

Drawing range is from 0,0 to 329,239. Coordinates are shown in the figure below.

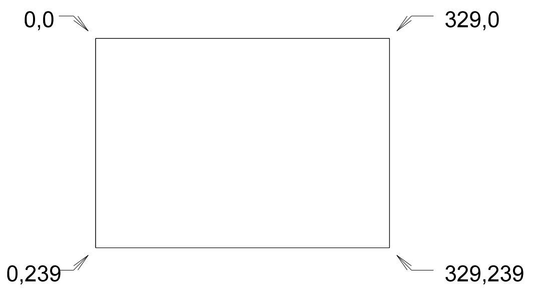


Figure 15-2

The demo program CIRCLE.BAS draws a circle using point set (DISPLAY P(x,y)).

Minimal range checking is performed. If a point does not show up or is in the wrong place, chances are either the X or Y parameter is wrong.

CLEAR, FILL, AND XOR AREAS

You can clear (erase), fill, or XOR (toggle) a rectangular area using one of the display commands.

DISPLAY F(x1,y1),(x2,y2)	Fills an area
DISPLAY F,C(x1,y1),(x2,y2)	Clears an area
DISPLAY F,X(x1,y1),(x2,y2)	Toggles an area

This command is useful when you want to selectively clear or fill an area (graphics, for example) without clearing the entire screen. Filling an area is useful for displaying a bar graph. It is faster than drawing lines.

XOR is useful when you want to highlight an area. The data is reversed in whatever area is selected.

The fill command takes time to execute, especially in large areas. This is important if you are running multitasking (ON COM, ON BIT, etc.).

Display fill operates fastest when X is on an 8 bit boundary. For example, filling an area bounded by (0,y),(96,y) is 20% faster than (1,y),(97,y), even though the area is the same. XOR requires an extra 20% more time to execute. Turning off sparkle (see earlier in this chapter) reduces write time by about ½.

Execution time depends upon the size of the area to fill. Larger areas obviously take more time than small ones.

LOAD AND SAVE SCREENS

Two commands load and save screen graphics.

DISPLAY LOAD (*x1*,*y1*),(*x2*,*y2*),*seg*,*address*
DISPLAY SAVE (*x1*,*y1*),(*x2*,*y2*),*seg*,*address*

The first syntax, DISPLAY LOAD, transfers graphic information from RAM or Flash to the screen. The second syntax, DISPLAY SAVE, transfers graphic from the display to RAM only.

address range is &0 to &FFFF.

Use DISPLAY LOAD to bring up pre-generated graphic icons (such as buttons), prompts, company logos, or process symbols. When used in conjunction with DISPLAY SAVE, it can be used to recall a graph

DISPLAY LOAD does not save any small text info.

Parameters (*x1*,*y1*),(*x2*,*y2*) specify the rectangular area on the display where the graphic will be placed.

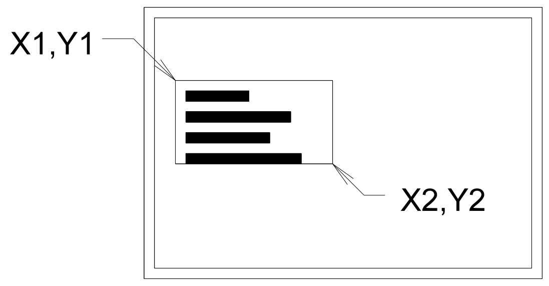


Figure 15-3

Graphic coordinates correspond to those used to draw lines, pixels, and fill areas.

The *seg* parameter specifies what memory segment to save or load.

Memory type & size	Range of *seg* parameter
128K RAM	1
512K RAM	1-7
128K Flash	8 -9
512K Flash	8-15

DISPLAY SAVE works only when *seg* is to RAM. To save a screen to flash, you must use SAVE.

You will have to calculate the *address*, especially if you have more than one screen.

Calculating graphic memory requirements

Use (*x1*,*y1*),(*x2*,*y2*) parameters in DISPLAY LOAD and SAVE to calculate the number of bytes needed.

bytes = ((X2 - X1) / 8) * (Y2-Y1)

The equation above assume X2 > X1 and Y2 > Y1. Round UP the result of (X2-X1)/8 if it is a fraction. See the following example.

Suppose you want to save screen data from (0,120) to (203,239).

bytes = ((203 - 0) / 8) * (239 - 120)
bytes = 25.375 * 119
(Round UP 25.375 to 26)
bytes = 26 * 119
bytes = 3094

To save multiple screens, assign each screen a number (assuming they are all the same size). When saving, multiply the screen number by the number of bytes/screen. This becomes the *address*.

Making and saving custom graphics
A program in the applications disk, BMP-GPH.EXE, translates a .BMP file into a hex file readable by the RPC-2350.

BMP-GPH converts a black and white, single layer (no transparent background color) BMP file generated by any number of PC graphics programs. Width and height attributes of the BMP graphic are in pixels. Limits are 320 pixels for width and 240 pixels for height.

Some BMP graphics are provided for your use. These are in the GRAPHICS directory of the application disk.

Graphics are transferred to the RPC-2350 using the HEXLOAD.BAS program, located in the FONT directory. Every download saves the graphic to the same location in RAM (segment 1, starting address 0). You must move the graphic to a predetermined location either in Flash or RAM.

Use the table at the end of this section to help map your graphics.

When saving to Flash, you must round UP the amount stored to the next Flash sector size. A 128K flash sector size is 128. A 512K flash sector size is 256. If you are unsure what you will use in the future, use the 256 byte sector size.

The following example shows how to determine the number of bytes to store to a sector into Flash.

Assume you have calculated your graphic memory requirement. For this example, we will use 3094 (from the previous example). Divide this number by 256

```
    3094 / 256 =  12.085
```

Rounding up 12.085 gives us 13. This is the number of sectors needed for saving to flash.

The number of bytes is determined by multiplying this number by sector size.

```
    13 * 256 =  3328
```

This graphic will use 3328 bytes of flash memory. If you are saving multiple graphics of the same size, you can use this number as an offset for each graphic, or picture, in the SAVE command.

```
    SAVE s,3328 * PIC,1,0,3328
```

For the above example, **s** is the segment number (8 or 9 or 8 to 15, depending on memory size), **PIC** is the picture number.

Since each segment = 65.536, you can hold 19 pictures in 1 segment.

Where you save graphics depends upon several factors, one of them being how you intend to use graphics. The second factor is the size of the graphics.

You will have to make a memory map of what you want to save to RAM and FLASH. Start by determining what you want to save. If you want to data log, set aside some memory for that. If you need to store a second program, allow for that. Don't forget that graphic fonts are stored in segment 9, address &AC00. If you are using the supplied fonts, you should not save data there.

Medium sized fonts are stored in segment 9 from address &AC00 - &B7FF. Large fonts are stored from &B800 to &FFFF. If you use one font, but not the other, you can write in the unused area.

ADDITIONAL SCREEN CONTROLS

The display is capable of other display modes. Some are listed here, others are in the SED1335 technical manual (File:M133XF.PDF). This is a complicated chip, and capable of many operating modes. We do not support programming this chip except as explained in this manual.

Changing cursor size or form
Default cursor type is a block type. This can be changed to an underline or smaller size.

Execute the following code to change the cursor width, height, and form.

```
    OUT   &F9,&5D
    OUT   &F8,width
    OUT   &F8,height OR form
```

width is in the range of 0 to 6. 0 gives a 1 pixel width.

form determines if it is an underline or block. Logically OR &80 to *height* for a block cursor.

height is in the range of 0 to 6. 0 gives a 1 pixel height.

Screen flashing and screen ON/OFF
CAMBASIC sets the cursor flash rate at 1/second. It can flash faster, but it is not as visible on the LCD display. You can also flash the character and graphics screens.

`DISPLAY ON` and `DISPLAY OFF` control the entire display or graphics and character screens individually. Since cursor and screen flashing are controlled by the same byte, you will have to control them as described below if you want to flash screens.

The parameter byte is made up of several components. Refer to the SED1335 PDF file, section 3.3.1, for detailed information. The following are parameter information you may need to know. The section number and name are given first, followed by an explanation.

3.3.1.1 D
This is bit 0 of the command. Writing a &58 to address &F9 turns off the display.

The following make up the parameter byte following the command. The command byte value is &59.

3.3.1.2 FC
FC1 and FC0 set the character cursor flash rate.

3.3.1.3 FP
Set the attributes of each screen. FP0 and FP1 are the character screen. FP2 and FP3 are the graphics layer, or screen. FP4 and FP5 are not used. Setting a flash rate at 16 Hz does a poor job of a half tone effect. A flash rate of 2 Hz does get attention.

Command example below sets the text layer to flash at a 2 Hz rate.

```
OUT &F9,&59
OUT &F8,10
```

This command turns on both the character and graphics layer.

```
OUT &F9,&59
OUT &F8,&17
```

This is equivalent to executing a DISPLAY ON command. Since text and graphics are or'ed (by default), you will see both.

These commands turn off the blinking cursor.

```
OUT &F9,&59
OUT &F8,&14
```

Screen overlay
This command selects how screen are displayed. There are two screens available. Layer 1 is character and layer 2 is graphics.

Pixels on each layer can be OR'ed, XOR'ed, AND'ed, and Priority-OR'ed. See Figure 44 in the SED1335 technical manual for more information. By default, all text and graphics are OR'ed.

Turn screens ON and OFF
The entire display, graphics, or small character screen display is turned off (blanked) by executing certain commands. This is useful when you want to alternate graphics and small character screen displays.

```
DISPLAY OFF
```

blanks the display. It is turned on by executing

```
DISPLAY ON
```

NOTE: Display ON/OFF does not control its power. See DISPLAY POWER ON/OFF below.

You can turn on or off graphics and character displays by entering the appropriate letter after ON or OFF.

```
DISPLAY OFF G
```

turns off the graphics screen.

NOTE: The cursor will continue to blink even if you turn off the display. Execute a **CONFIG DISPLAY 0,8,0** before writing to the display to turn off the cursor. You can manually control the cursor and screens. See "Screen flashing and screen ON/OFF" above.

Display power ON/OFF
The display can enter a power off mode (the controller manual calls it "Sleep In"). This mode shuts off the controller signals. Bias power to the display is supposed to be removed. However, it is not on the RPC-2350. Bias power cannot be removed since is supplies power to RS-232, analog input and outputs. Therefore, do not enter a display power OFF mode.

Some power can be saved by turning off the LCD back light. The ground lead must be connected to P2 terminal marked "SWPWR:. On/off control is performed by executing the following command:

```
OUT &E7,1
```

To turn the inverter off, execute

```
OUT &E7,0
```

Display mapping
The following information is for programmers who understand how the SED1335 chip works and want to put it into modes not supported by CAMBASIC.

The following is the memory map sent to SCROLL register in the SED1335..

Layer 1 (character)	SAD 1 &0000
Layer 2 (graphics)	SAD 2 &2000
Layer 3 (graphics)	SAD 3 &6000
	SAD 4 &A000

PRINT AND DISPLAY TIMES

Execution times for a command (not character) shown below.

Command	Time (in milli-Sec)	Condition
DISPLAY A$;CHR$(13);	1.255	A$ is 20 characters long
DISPLAY (x,y) A$;	1.2	A$ is 20 characters long
DISPLAY A$	62	Timed when cursor was at bottom of screen. This is scroll up time.
DISPLAY F(x1,y1),(x2,y2)	80	Area to clear or fill is 96 x 96, on even boundary, sparkle off.
DISPLAY F, X(x1,y1), (x2,y2)	100	XOR area is 96 x 96, sparkle off.
DISPLAY M (x,y) A$;	40	A$ is 10 characters long, no sparkle mode.
DISPLAY M (x,y) A$;	16	A$ is 10 characters long, allow sparkle
DISPLAY L (x,y) A$;	80	A$ is 5 number long, no sparkle mode
DISPLAY L (x,y) A$;	10	A$ is 5 numbers long, allow sparkle
DISPLAY P(x,y)	0.425	Same x,y point
DISPLAY LINE (0,0),(10,10)	8	14 points in line

Times were calculated in CAMBASIC by dividing executing time by the number of loops. General program was in the form of:

```
10 CLEAR TICK(0)
20 FOR N = 0 TO 999
30 DISPLAY test
40 NEXT
50 PRINT TICK(0)/1000
```

Times shown are what it took to execute the entire DISPLAY command.

Reverse medium and large characters does not add a significant amount of time.

EL DISPLAY

An EL display from Planar (Model EL320.240.36) may be connected to the RPC-2350G. + 5V and + 12V power must be available. + 12V is connected to P2, "ELPWR" pin. + 5V is taken from the board.

+ 12V may also be used to supply the board. Simply connect "ELPWR" on P2 to "7-30V" on P2. Install jumper W8.

To use your own + 5V supply, connect it to the "+ 5V" pin on P2. Make sure jumper W8 is removed. Do not connect "ELPWR" to any other pin on P2.

The EL display has very fast pixel on/off time. Consequently, sparkle is more noticeable even when it is off (default). Sparkle is more noticeable when an area is illuminated.

Programming the EL display is the same as LCD.

EL display connects to J13. Connection is one to one using a 2 mm ribbon cable. Cable is available from Samtec (www.samtec.com or 812 944 6733), part number TCSD-10-D-12.00-01-F

CABLE PIN OUTS

The following tables are cable pin outs for LCD and EL displays. J9 is the 20 pin display connector on the board.

J9 Pin #	Description	LCD pin #
1	FLM	1
2	LP	2
3	CP	3
4	WF	4
5	Contrast adjust	5
6	+ 5V	6
7	Ground	7
8	Minus bias (≈-22V)	8
9	XD0	9
10	XD1	10
11	XD2	11
12	XD3	12
13	Display on/~off	13
14	no connection	14
15-20	no connection	

J13 Pin #	Description	EL pin #
1	Display + 12V	1
2	Display + 12V	2
3	Sef test (no connect)	3
4	Reserved	4
5	+ 5V power	5
6	Ground	6
7	FLM/VS	7
8	Ground	8
9	LP/HS	9
10	Ground	10
11	CP/VCLK	11
12	Ground	12
13	XD0	13
14	Ground	14
15	XD1	15
16	Ground	16
17	XD2	17
18	Ground	18
19	XD3	19
20	Ground	20

COMMANDS

The following commands are used with the graphics display:

Command	Description
CLEAR DISPLAY	Clear graphics and character displays.
CLEAR DISPLAY C	Clear character screen only.
CLEAR DISPLAY G	Clear graphics screen only.
CLEAR DISPLAY LINE	Clear small text line at current cursor row.
CLEAR DISPLAY LINE (x1, y1),(x2, y2)	Clear graphics line from (x1,y1) to (x2,y2)
CLEAR DISPLAY P(x,y)	Clear graphics point
CONFIG DISPLAY	Configure display and cursor type
DISPLAY "text"	Print and optionally format text and numbers.
DISPLAY (row, col)	Position cursor for writing small characters.

Graphics memory map table.

Description	Size	Start address	End address
Sample	*3100*	*1:0*	*1:&c0D00*

Command	Description
DISPLAY F(x1,y1),(x2,y2)	Fill area bounded by (x1,y1) to (x2,y2)
DISPLAY F,C(x1,y1),(x2,y2)	Clear area bounded by (x1,y1) to (x2,y2)
DISPLAY F,X(x1,y1),(x2,y2)	XOR area bounded by (x1,y1) to (x2,y2)
DISPLAY ON [type]	Turn character, graphic, or both screens on
DISPLAY OFF [type]	Turn character, graphic, or both screens off
DISPLAY P(x,y)	Turns a graphic point on or off
DISPLAY LINE	Draws or erases a graphic line.
DISPLAY M (x,y)text	Print medium characters
DISPLAY M, R(x,y)text	Print medium characters in reverse video
DISPLAY L (x,y)text	Print large characters.
DISPLAY L,R(x,y)text	Print large characters in reverse video

CHAPTER SYNOPSIS

- Power input options
- Power output options
- Expansion port description
- Expansion port pin out

POWER INPUT

There are different power I/O options for the RPC-2350 and RPC-2350G. Read this carefully to determine which one applies to your board.

The RPC-2350 and RPC-2350G have different input power options. The RPC-2350 operates from + 5, ±0.25VDC only.

The RPC-2350G operates from either + 5, ±0.25VDC or + 6.5 to + 30V DC. To operate from 6.5 to 30V, make sure jumper W8 is installed. This connects + 5V regulated output to the board.

If you intend on using + 5V power out, be sure to read "Heat sinking" below for related information.

+ 5V input current, for budgetary purposes, is about 250 Ma. The RPC-2350 uses slightly less power.

Both models generate voltages for RS-232 and analog outputs are generated on card. The RPC-2350G generates a regulated negative bias voltage for the LCD display.

Power is applied at P2. See Figure 16-1 for location.

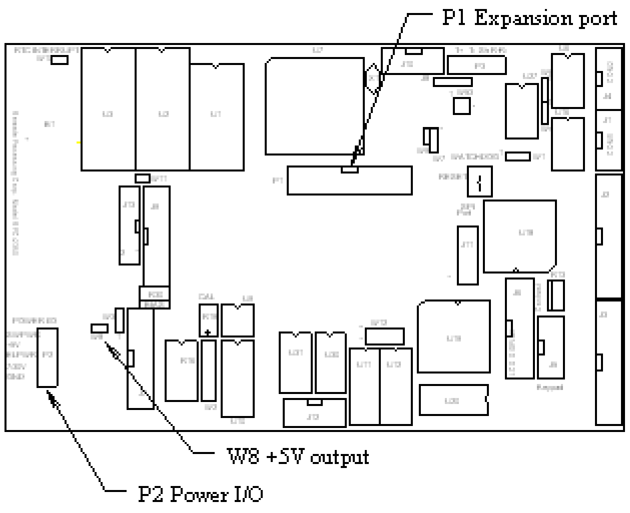


Figure 16-1 Power, Expansion, and jumper locations

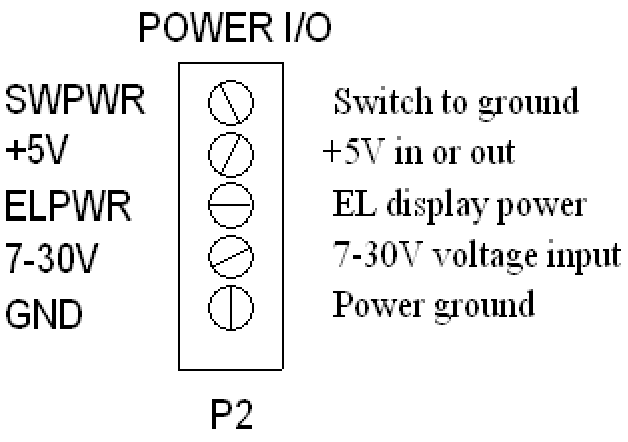


Figure 16-2 Power connector detail

POWER OUTPUT

There are several power outputs available from both RPC-2350 and RPC-2350G. ±12V power is available at the analog connector J7. Current is limited to about 40 Ma. See *Chapter 8,* "ANALOG POWER SUPPLY" for more information.

The RPC-2350G can supply regulated + 5V power at P2, with 6.5 to 30V at the input. Approximately 750 Ma. is available for external devices. See "Heat sinking" below if you intend to draw any significant (> 200 mA.) current from the board.

This extra current can supply EL displays, opto racks, LED displays and back lighting, or other devices.

Heat sinking
A heat sink under U23 is normally not necessary. You should use a heat sink when all 4 conditions below are met:

1. You are supplying 6 to 30VDC to power the board.
2. Current from the board (to power external devices such as opto racks) approaches 750 mA.
3. The ambient temperature will be more than 60° C.
4. Supply voltage is usually less than 9 volts.

WARNING:

U23 can get ***VERY HOT***, exceeding 100°C (hotter than boiling water) and still operate normally. DO NOT TOUCH U23!

Normally U23 is very warm to touch (40°C). As current demand increases and/or supply voltage decreases, its temperature increases.

U23 uses a heat sink for a TO-220 IC. Suggested heat sink by Aavid is: 577202B00000. This part is available from DigiKey (800 344 4539).

EXPANSION PORT P1

The expansion port brings out address, data, and control lines for an external board. This external board can consist of counters, timers, digital I/O, and analog I/O. 6 address and 8 data, read, write, select, and other power and control lines are provided.

Expansion port I/O address range is &100 - &13F. This equals 64 addresses.

Timing is simple, based on Z80 I/O signals. Access times for external devices should be 200 nS. or faster. Data is read on the rising edge of IRD. See timing diagram below.

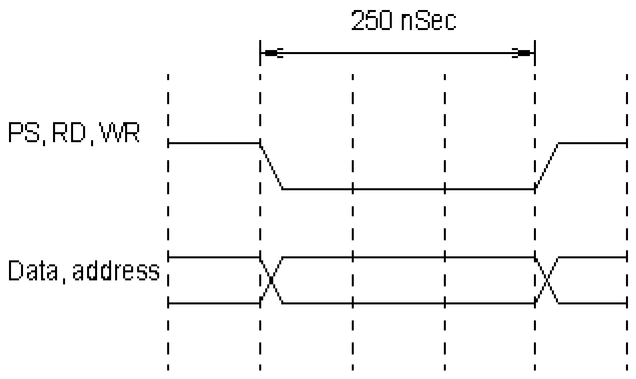


Figure 16-2 I/O timing

All logic levels are TTL. High speed CMOS IC;s should be used when interfacing to this bus.

Connection between P1 and your board is via a simple 26 pin ribbon cable. Limit cable length to 3 inches.

Expansion port connector pin out is listed below.

P1 pin #	Description
1	Data 0
2	Expansion port address select
3	Data 1
4	6.5 to 30V power (P2 “7-30V” pin)
5	Data 2
6	NMI
7	Data 3
8	+ 5V power
9	Data 4
10	Reset (active low)
11	Data 5
12	Ground
13	Data 6
14	Ground
15	Data 7
16	Address 4
17	Address 0
18	+ 5V power
19	Address 1
20	Address 5
21	Address 2
22	Push button reset (low)
23	Address 3
24	nc
25	I/O read (active low)
26	I/O write (active low)

CHAPTER 17 RESOURCES

CHAPTER SYNOPSIS

● List sources of accessory components

The RPC-2350 can be used in a number of applications. We provide common accessories. However, your application probably requires something we do not stock.

Below is a listing of some components you may use with the RPC-2350. This is not an extensive listing by any means. Preference is given to companies that you can order parts from distributors.

NOTE: The following list is provided as a service. REMOTE PROCESSING MAKES NO WARRANTY AS TO FITNESS FOR A PARTICULAR USE. We have not even touched most of these products, so we don't know if it will work with ours.

We will work with you for what we deem a reasonable amount of time to get a particular product to work with our board. We have a limited knowledge of these products so we can't promise you good support.

RESOURCE LISTING

The following list was compiled in May, 1999. Companies phone numbers (such as area codes) may have changed, products discontinued, or the company stopped operations since this time.

Part distributors
The following distributors are mentioned as potential suppliers of parts listed below. Web addresses and phone numbers are believed accurate as of October, 2004.

Didi-key:
www. digikey.com
800-344-4539

Allied Electronics:
www.alliedelec.com/
800-433-5700

VF displays
IEE displays, California. Ph: 818 787 0311
www. ieeinc.com

Makes vacuum florescent and LCD displays. Some products available from Allied Electronics. Century series are driven through a serial port.

Alternative LCD's
IEE displays, California. Ph: 818 787 0311
www. ieeinc.com

Makes large size (0.75") and extended temperature range (to -20°C) displays. Some available from Allied Electronics.

Apollo Displays in New York supplies a wide range of Optrex displays. PH: 516 654 1143
www. apollodisplays.com

PWM motor drivers
Solutions Cubed in California makes the Motor Mind B. It features a serial interface to adjust DC motor speed and direction. Current draw is 2A continuous, 30V maximum.
Ph: 530 891 8045
Fax: 530 891 1643
www. solutions-cubed.com

Stepper Motor Control and Driver
There are a number of motor controller suppliers. Below are ones available through distribution.

Modern Technology provides one through Digi-key. Digi-key part number MTSD-V1-ND

Gallial makes an extensive line of motor controllers. Contact factory at 650 967 1700

CyberPack Co. Makes stepper motor driver. 630 493 0954 www. cyberpakco. com

Haydon Switch & Instrument Inc. Stepper motors and drivers. 203 756 7441 www. hsi-inc.com

OPTO modules
The MPS-XX series boards accept OPTO-22 G4 series or Grayhill G5 modules. Please keep in mind our boards are designed to plug straight into the RPC-2350 board with a simple ribbon cable. Both Opto-22 and Grayhill supply boards to plug their modules into but they will not easily work with ours.

Opto-22 G4 modules are available from Allied Electronics.

Grayhill G5 modules are available from Digi-key.

Isolated RS-485 and communications
B&B Electronics in Illinois. Ph: 815 433 5100
Makes several communication and isolation products.

RS-232 to RS-485 converters
B&B Electronics in Illinois. Ph: 815 433 5100
Makes several converter products.

Octagon Systems, Colorado Ph: 303 430 1500. The MTB-485 takes RS-232 from your PC and outputs RS-485. Very easy to use. Have used this product on our boards. We support the MTB-485 with our boards.

Temperature sensors and transmitters
Minco, Minnesota Ph: 612 571 3121
Assortment of RTD's, thermocouples, and transmitters.

ELECTRICAL

CPU
Z8S180, 18.432 MHZ clock

Memory
CAMBASIC, 32K ROM (U1). Mapped per below.

Programming and data is 128K RAM (U2). Expandable to 512K by installing part number 1039.

Programs are stored in 128K Flash type EPROM (U3). Expandable to 512K by installing part number 1301.

Digital I/O
The RPC-2350 has 48 digital I/O lines from 82C55 IC. 24 are from J2, which is a general purpose port. The other 24 are from J3. J3 has 8 high current outputs, which may be jumpered for inputs. Keypad port J5 uses 8 of the 24 lines. LCD display port J6 uses another 8.

The specifications below are for all digital I/O except for the eight high current lines at J3.

Drive current	2.5 mA. maximum per line, sink or source. TTL compatible.
Output low voltage	0.45V max at 2.5 mA, 1V max at 15 mA. for opto rack.
Output high volts	2.4V minimum, sink or source at rated current.

All digital input lines are TTL compatible.

High current output at J3
8 of the 24 lines can drive up to 500 mA. at 50V. Refer to *CHAPTER 6, DIGITAL AND OPTO PORTS* for limitations.

Analog input(A/D)

Channels:	8
Ranges:	0-5V, ±2.5V
Resolution:	12 bits (4096 counts)
Accuracy:	±3 counts + 3 counts offset
Type:	Single ended or pseudo differential

Conversion time: 500 micro-seconds in CAMBASIC

Analog voltage output(D/A) (option)

Channels:	2
Ranges:	0-5V, ±5V, 0-10V
Resolution:	12 bits (4096 counts)
Accuracy:	±3 counts + 3 counts offset
Current output:	5 mA. for accuracy
Settling time:	2 micro-seconds

4-20 mA. output(option)

Channels:	2 (each requires its own D/A), non-isolated
Input voltage:	12V to 30V or use on card supply (≈15V available)

Keypad input
10 lines accept a 16 to 24 position matrix keypad. Scanning and debounce performed in CAMBASIC. Uses 8 lines from J3. 24 position uses additional 2 lines from J3.

Serial ports
Two RS-232D serial ports. COM1 has TxD and RXD only. COM2 also has RTS and CTS lines. Programmable baud rates from 600 to 38.4K, 7 or 8 data bits, parity even, odd, or none, 1 or 2 stop bits.

Flash EPROM
Accepts Atmel 29C010A, 29C040A or equivalent PEPROM.
Size:128K (29C010A) or 512K (29C040A)
Speed:120ns or faster.

Watchdog timer, reset
Watch dog timer resets card for 150 ms minimum when enabled. Push button reset included.

Input power
+ 5VDC ±5% at 375 ma.
+ 12V at 250 mA. (RPC-2350G)

Current draw for 7-30V input depends upon applied voltage. 7V supply draws more current (about 420 mA) than 12V (about 200 ma).

Current consumption does not include any opto-modules or other accessories.

Output power

+ 5V at P2:	750 mA. with 7 to 30V input at "7-30V" terminal on P2, W8 jumpered.
±12V at J7	Up to 40 mA. @ + 12, 20 mA. @ -12V Note: Subtract any current used by analog outputs.

Environmental
Temperature range: -20°C to 70°C. Temperature can go slightly higher if air flows across board.

Humidity: 0% to 95%, non-condensing. For increased battery life, some humidity should be present.

Vibration: 5 g's maximum, 5Hz to 500 Hz, each axis.

MECHANICAL

Size: 5.0" x 8.0"
Maximum height: 0.6", with ribbon cable installed, no strain relief.

Mounting holes: 4 each corner. Hole size is 0.175" dia. See drawing at end of this chapter.

MEMORY AND I/O MAP

Memory maps are with respect to the CPU, not CAMBASIC.

Memory

Description	Address
CAMBASIC U1	&00000 - &06FFF (RPC-2350G) &00000 - &05FFF (RPC-2350)
RAM, U2	&07000 - &1FFFF (RPC-2350G) &06000 - &1FFFF (RPC-2350) - &7FFFF all w/ 512K
Flash U3	&80000 - &9FFFF all w/128K &80000 - &FFFFF all w/ 512K

I/O

Description	Address
J2 Digital	&000 - &003
J3 Digital	&040 - &043
J5 Keypad	&042
Internal processor	&0080 - &00BF
Analog output 0	&0C0 - &0CF
Analog output 1	&0D0 - &0DF
Watchdog	&0E4 - &0E4
Counter	&0F0 - &0F7
Graphics display	&0F8 - &0FF
Expansion port	&100 - &13F
SPI port	&E0 SCLK &E1 SPI in (to device) &E5 Select 1 &E6 Select 2 &E8 SPI out (from dev.)

JUMPER DESCRIPTIONS

A * after a jumper position indicates factory default and is jumpered.

Jumper	Description
W1[2-3]*	Watchdog timer 1.2 Seconds
W1 open	Watchdog timer 150 mS
W1[1-2]	Watchdog timer 1.2 seconds
W2[1-2]*	128K RAM. Flash not affected
W2[4-5]*	128K Flash. RAM not affected
W2[2-3]	512K RAM. Flash not affected
W2[5-6]	512K Flash. RAM not affected
W3[2-3]*	Manual contrast control for Graphic LCD display
W3[1-2]	Software contrast control for Graphic LCD display
W4[2-3]*	COM2 RS-232/485 select at RS-232
W4[1-2]	COM2 RS-232/485 select at RS-485
W5[1-2][3-4]*	RS-485 network terminator.
W6[1-2]*	RS-485 in 2 wire mode (Receive off when transmitting)
W6[2-3]	RS-485 is 4 wire mode (Receive always on)

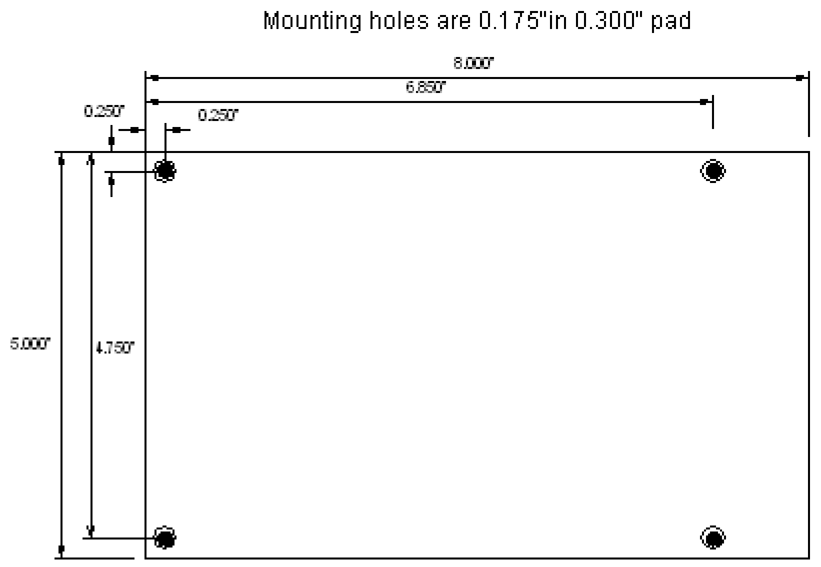
Jumper	Description
W7[1-2]*	INT1 to counter carry
W7[2-3]	INT1 to counter borrow
W8[1-2]	Regulated + 5V output to board + 5V layer.
W9[1-2]	High voltage interface to counter
W10[1-2]	Real time clock interrupt output to INT0
W11[1-2]*	Enable autorun and graphics fonts
W12[2-4]*	D/A output 0 to 0-5V
W12[1-3]*	D/A output 1 to 0-5V
W12[8-10]	D/A output 0 to 0-10V
W12[7-9]	D/A output 1 to 0-10V
W12[6-8]	D/A output 0 to ±5V
W12[5-7]	D/A output 1 to ±5V

W13 provides pads for D/A input filter capacitors.

CONNECTOR DESCRIPTION

The following table provides a brief function description for each connector and the chapter number where you can find information.

Connector Desg.	Function	Chapter(s) for more info.
P1	Expansion port	16
P2	Power I/O	6,16
P3	RS-422/485 port	4
J1	COM 1 RS-232	4
J2	Digital I/O	6
J3	Digital I/O (shared)	6
J4	COM 2 RS-232	4
J5	Keypad	6,9
J6	LCD character display	6,10
J7	A/D and D/A	8
J8	RS-422/485 port	4
J9	LCD graphic display	15
J10	Counter and interrupt input, pulse outputs	11,13,14
J11	SPI port	4
J12	4-20 mA output	8
J13	EL graphic display	15



RPC-2350 board outline and mounting hole locations